



Caching Across Enterprise Application Tiers

Reza Rahman

Expert Group Member, Java EE 6 and EJB 3.1

Resin EJB 3.1 Lite Container Developer

Author, EJB 3 in Action

reza@caucho.com



Caching

- **Caching is an essential tool for scaling web applications**
- **The basic concept is to avoid disk access, I/O, CPU power and network traffic – primarily to the backend relational database in modern web applications**
- **With memory becoming a commodity, it makes more and more sense to make intelligent use of caches – the real bottleneck is synchronization across the network**
- **Across enterprise application tiers, caching comes in many forms, some obvious and others not so obvious**

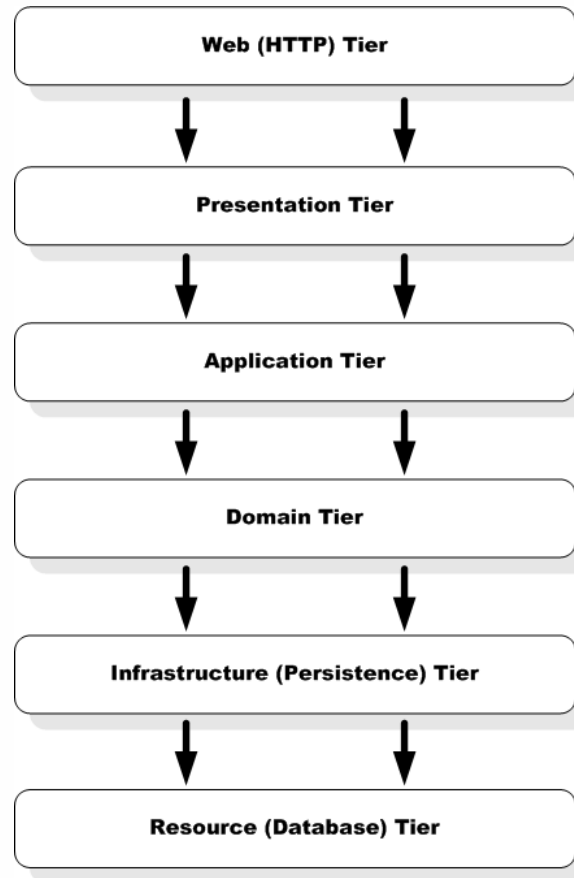


Enterprise Application Tiers

- **Layering is an essential way of managing complexity for non-trivial, team based (often distributed team based in a post-outsourcing world) enterprise applications**
- **Traditional layered architectures (a la J2EE 1.4 design patterns) and domain driven design segment enterprise applications into roughly four tiers**
- **Even a minimalist approach (a la Seam) utilizes specialist APIs per logical tier**
- **Each of the tiers has caching of its own flavor**



Enterprise Application Tiers





Web (HTTP) Tier

- **Web (HTTP) tier caching is the easiest, most non-invasive and a very effective way of enhancing application performance**
- **You should consider caching static content such as straight HTML, images, PDF files, CSS and JavaScript**
- **You should also consider caching dynamic content that changes only infrequently**
- **The HTTP protocol provides robust support for caching**



HTTP Caching Header Example

```
HTTP/1.1 200 OK
Date: Fri, 30 Oct 1998 13:19:41 GMT
Server: Apache/1.3.3 (Unix)
Cache-Control: max-age=3600, must-revalidate
Expires: Fri, 30 Oct 1998 14:19:41 GMT
Last-Modified: Mon, 29 Jun 1998 02:28:12 GMT
ETag: "3e86-410-3596fbbc"
Content-Length: 1040
Content-Type: text/html
```

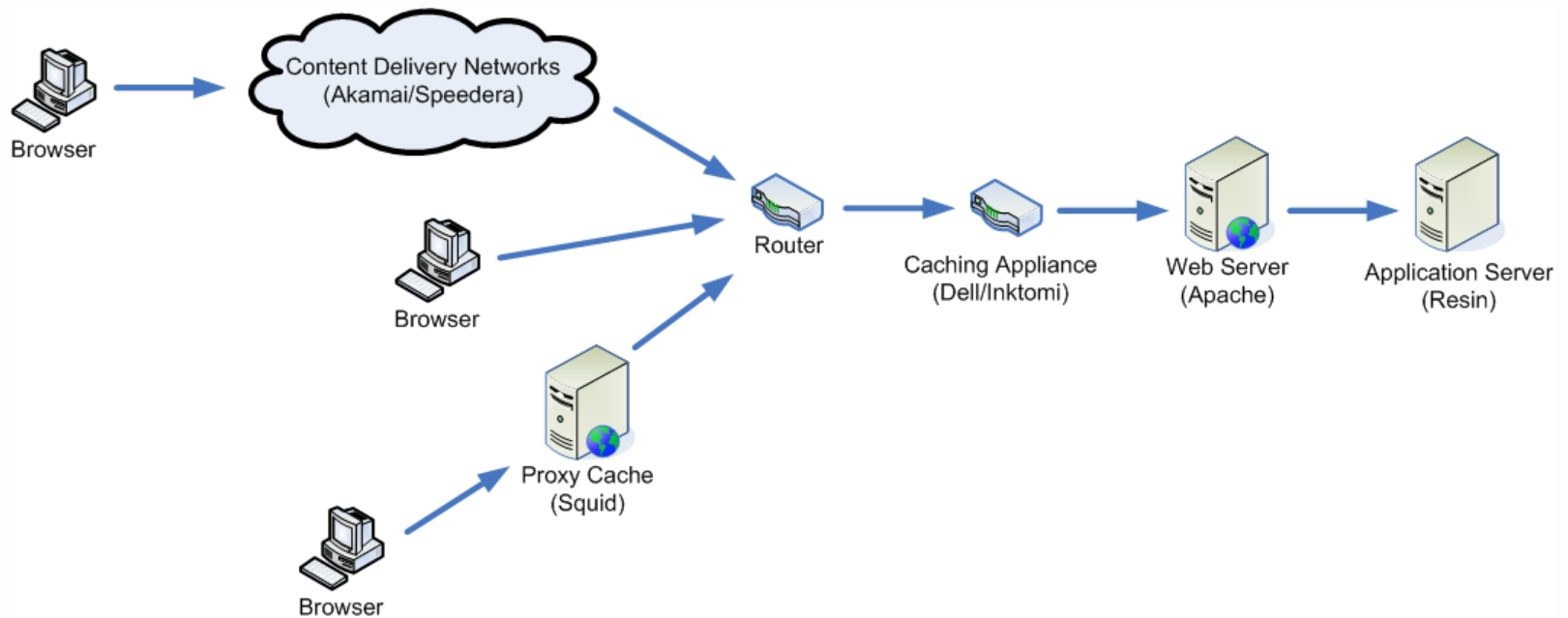


HTTP Caching Headers

HTTP Header	Description
Last-modified	The time the content was last changed
ETag	Unique identifier for a given version of content
Expires	Time when content expires
Cache-Control	Explicit instructions for caching: <i>public</i> (definitely cache), <i>private</i> (browser cache only), <i>no-cache</i> (don't cache), <i>max-age=n</i> (set expiration), <i>s-maxage=n</i> (set expiration for proxy caches only), <i>must-revalidate</i> (no short-cuts for validation), <i>proxy-revalidate</i> (no short-cuts for validation for proxies only)



Web Caching in the Data Center





Apache Web Server Caching

```
### activate mod_expires
ExpiresActive On
### Expire .gif's 1 month from when they're accessed
ExpiresByType image/gif A2592000
### Expire everything else 1 day from when it's last
modified
### (this uses the Alternative syntax)
ExpiresDefault "modification plus 1 day"
### Apply a Cache-Control header to index.html
<Files index.html>
Header append Cache-Control "public, must-revalidate"
</Files>
```



Resin Proxy Caching

```
<resin xmlns="http://caucho.com/ns/resin">
  <cluster id="web-tier">
    <cache entries="16384"
      disk-size="2G"
      memory-size="256M" />
    <server id="a" address="192.168.0.10" />
    <host host-name="www.foo.com">
  </cluster>
</resin>
```



Programmatic Cache Control

```
<%@ page session="false" %>
<%! int counter; %>
<%
long now = System.currentTimeMillis();
response.setDateHeader("Expires", now + 15000);
%>
Count: <%= counter++ %>
```



Presentation Tier

- **Rich application state maintenance natural caching mechanism - avoids needless database interaction (a la old school CGI/PERL/PHP or some action-oriented web frameworks)**
- **Correct granularity (scope) is critical for efficient memory use**
- **Servlets/JSP, JSF, CDI (or Spring) provide roughly expanding granularity and abstraction**
- ***Activation/passivation critical for proper memory management***
- ***Most application servers automatically cluster state***



Web Tier Scopes

API	Scopes
Servlet/JSP	Application (Servlet Context), session, request, cookies
JSF	@ApplicationScoped, @SessionScoped, @ViewScoped, @RequestScoped, Flash object
CDI	@ApplicationScoped, @SessionScoped, @ConversationScoped, @RequestScoped (more scopes such as <i>named conversations</i> and <i>window scope</i> available through CDI plug-ins)
Spring	Singleton, global session, session, request, prototype (Spring Web Flow provides some expanded scopes such as conversations)



Saving State in Servlets

```
request.setAttribute("request-key", "request-value");

HttpSession session = request.getSession(false);

if (session != null) {
    session.setAttribute("session-key",
        "session-value");
}

getServletContext().setAttribute("application-key",
    "application-value");

response.addCookie(
    new Cookie("cookie-key", "cookie-value"));
```



JSF Page with CDI

```
<h:form>
  <h1>Login</h1>
  <h:panelGrid columns="2">
    <h:outputLabel for="username">
      Username:</h:outputLabel>
    <h:inputText id="username"
      value="#{credentials.username}"/>
    <h:outputLabel for="password">
      Password:</h:outputLabel>
    <h:inputSecret id="password"
      value="#{credentials.password}"/>
  </h:panelGrid>
  <h:commandButton value="Login"
    action="#{login.login}"/>
</h:form>
```



CDI JSF Model

`@Named`

`@RequestScoped`

```
public class Credentials implements Serializable {  
    private String username;  
    private String password;  
  
    ... Getters and setters ...  
}
```




CDI JSF Event Handler

```
@Named @SessionScoped
public class Login implements Serializable {
    @Inject
    private Credentials credentials;
    @Inject
    private UserService userService;
    private User user;

    public String login() {
        user =
            userService.getUser(credentials.getUsername());
        ... Do a password check here...
        return "account.xhtml";
    }

    @Named @Produces @LoggedIn
    public User getCurrentUser() {
        return user;
    }
}
```



CDI Web Tier Producer

```
public class AccountManager implements Serializable {
    @Inject @LoggedIn
    private User user;

    @Inject
    private AccountService accountService;

    @Named
    @Produces
    @SessionScoped
    @SelectedAccount
    public Account getCurrentAccount()
    {
        return accountService.getAccount(
            user.getUsername());
    }
}
```



JSF Workflow (Step 1)

```
<h1>From Bank Account</h1>
<h:panelGrid columns="2">
  <h:outputLabel for="fromBank">
    Bank Name:</h:outputLabel>
  <h:inputText id="fromBank"
    value="#{transfer.fromBank}" />
  <h:outputLabel for="fromAccount">
    Account Number:
  </h:outputLabel>
  <h:inputText id="fromAccount"
    value="#{transfer.fromAccount}" />
</h:panelGrid>
<h:commandButton value="Start"
  action="#{transfer.enterToBank}" />
```



JSF Workflow (Step 2)

```
<h1>To Bank Account</h1>
<h:panelGrid columns="2">
  <h:outputLabel for="fromBank">From Bank:</h:outputLabel>
  <h:outputText id="fromBank"
    value="#{transfer.fromBank}"/>
  <h:outputLabel for="fromAccount">
    From Account:
  </h:outputLabel>
  <h:outputText id="fromAccount"
    value="#{transfer.fromAccount}"/>
  <h:outputLabel for="toBank">Bank Name:</h:outputLabel>
  <h:inputText id="toBank" value="#{transfer.toBank}"/>
  <h:outputLabel for="toAccount">
    Account Number:
  </h:outputLabel>
  <h:inputText id="toAccount"
    value="#{transfer.toAccount}"/>
</h:panelGrid>
<h:commandButton value="Next" action="enter_amount.xhtml"/>
```



JSF Workflow (Step 3)

```
<h1>Transfer Amount</h1>
<h:panelGrid columns="2">
  <h:outputLabel for="fromBank">From Bank:</h:outputLabel>
  <h:outputText id="fromBank" value="#{transfer.fromBank}"/>
  <h:outputLabel for="fromAccount">
    From Account:</h:outputLabel>
  <h:outputText id="fromAccount"
    value="#{transfer.fromAccount}"/>
  <h:outputLabel for="toBank">From Bank:</h:outputLabel>
  <h:outputText id="toBank" value="#{transfer.toBank}"/>
  <h:outputLabel for="toAccount">To Account:</h:outputLabel>
  <h:outputText id="toAccount" value="#{transfer.toAccount}"/>
  <h:outputLabel for="amount">Amount:</h:outputLabel>
  <h:inputText id="amount" value="#{transfer.amount}"/>
</h:panelGrid>
<h:commandButton value="Finish" action="#{transfer.transfer}"/>
```



CDI Conversations

```
@Named @ConversationScoped
public class Transfer implements Serializable {
    @Inject
    private transient AccountService accountService;
    @Inject
    private Conversation conversation;
    private String fromBank;
    private String fromAccount;
    private String toBank;
    private String toAccount;
    private double amount;

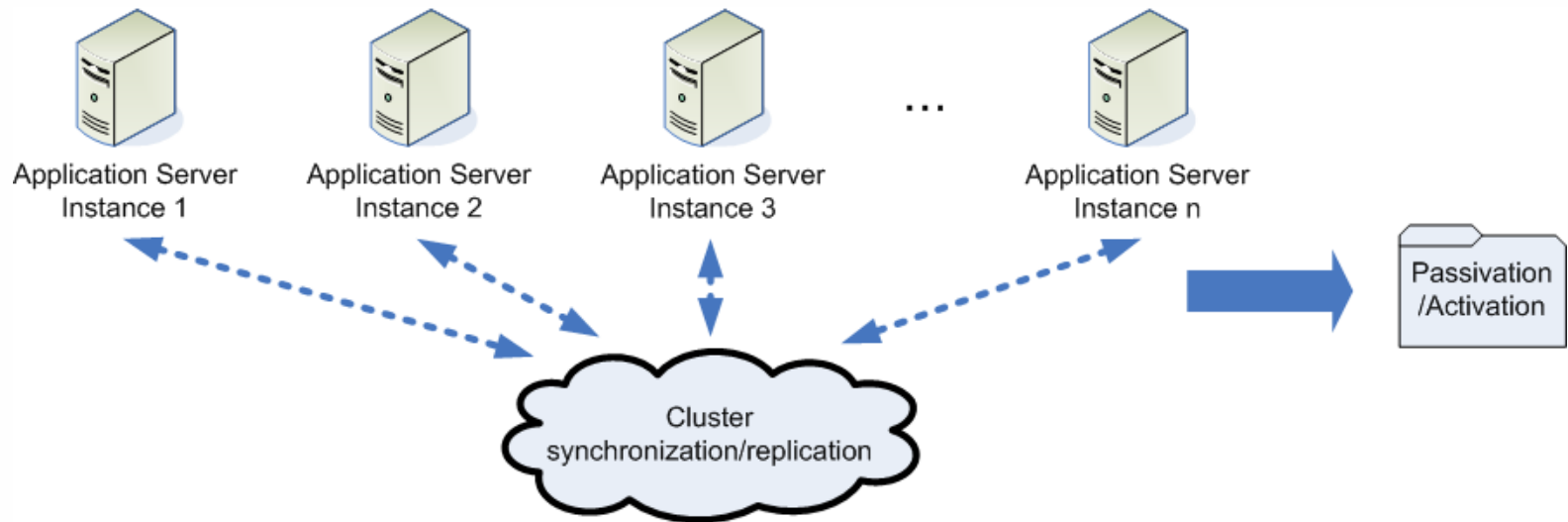
    ... Getters and setters ...

    public String enterToBank() {
        conversation.begin();
        return "enter_to_bank.xhtml";
    }

    public String transfer() {
        accountService.transfer(toBank, toAccount, fromBank,
            fromAccount, amount);
        conversation.end();
        return "account.xhtml";
    }
}
```

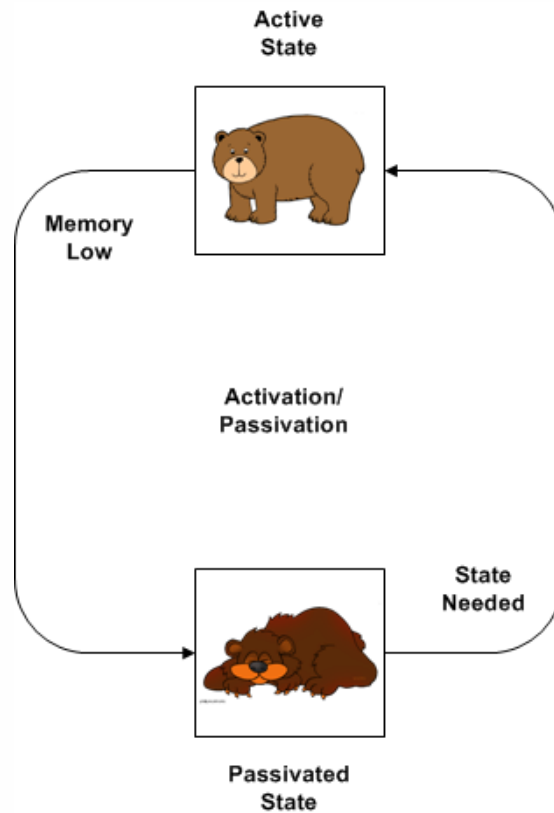


Server State Behind the Scenes





Activation/Passivation





Application Tier

- **CDI's (or Spring's) scoping, context and state management capabilities can be used in the application tier as well**
- **EJB stateless beans/MDB offer pooling**
- **EJB stateful beans offer better support for activation/passivation as well as persistence caches**
- **EJB singleton beans offer better support for concurrency**
- **Terracotta for Spring provides EJB-like clustering**
- ***We'll talk about direct distributed caching tools later***



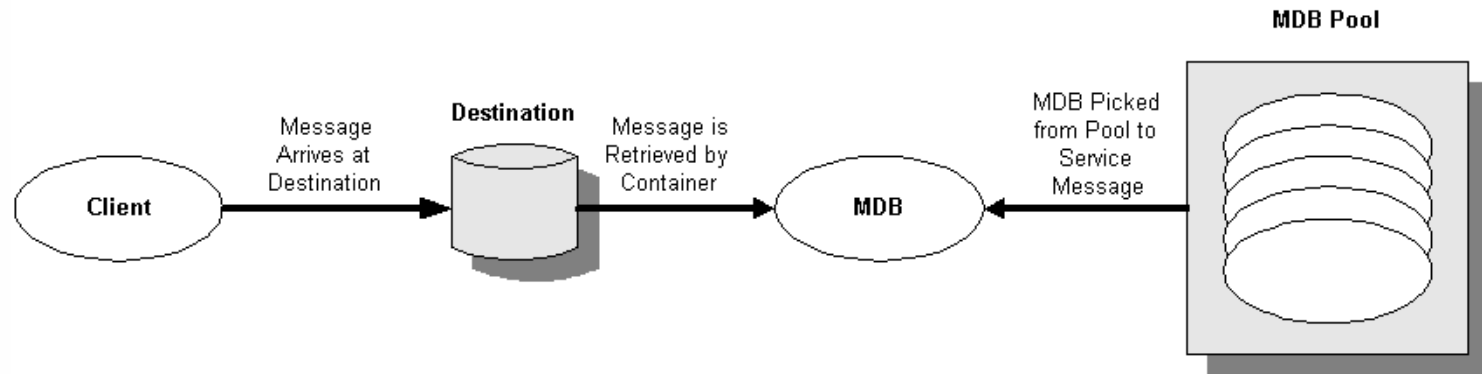
Message-Driven Bean

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(
        propertyName="destinationName",
        propertyValue="jms/OrderBillingQueue"))
public class OrderBillingMDB implements MessageListener {
    ...
    public void onMessage(Message message) {
        try {
            ObjectMessage objectMessage = (ObjectMessage) message;
            Order order = (Order) objectMessage.getObject();

            try {
                bill(order);
                notifyBillingSuccess(order);
                order.setStatus(OrderStatus.COMPLETE);
            } catch (BillingException be) {
                notifyBillingFailure(be, order);
                order.setStatus(OrderStatus.BILLING_FAILED);
            } finally {
                update(order);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    ...
}
```



Message-Driven Bean Pooling





EJB Stateful Bean

`@Stateful`

```
public class BidderAccountCreatorBean
    implements BidderAccountCreator {
    @Resource(name="jdbc/ActionBazaarDS")
    private DataSource dataSource;
    private Connection connection;
    ...
    @PostConstruct @PostActivate public void openConnection() {
        ...
        connection = dataSource.getConnection();
        ...
    }
    ...
    @PrePassivate @PreDestroy public void cleanup() {
        ...
        connection.close();
        ...
    }
}
```



EJB Singleton Bean

```
@Singleton @Startup
public class SharedDataBean {
    @PersistenceContext
    private EntityManager entityManager;
    private Data data;

    @PostConstruct private void init() {
        data = entityManager.find(Data.class, 1);
    }

    @PreDestroy private void destroy() {
        entityManager.merge(data);
    }

    @Lock(WRITE) public void setData(Data data) {
        this.data = data;
    }

    @Lock(READ) public Data getData() {
        return data;
    }
}
```



Extended Persistence Context

```
@Stateful
public class ShoppingCartBean implements ShoppingCart {
    @PersistenceContext(type=EXTENDED)
    private EntityManager entityManager;
    private Order order;
    ...
    public void createOrder() {
        order = new Order();
        ...
        entityManager.persist(order);
    }

    public void addToCart(Item item) {
        order.addItem(item);
    }
    ...
}
```

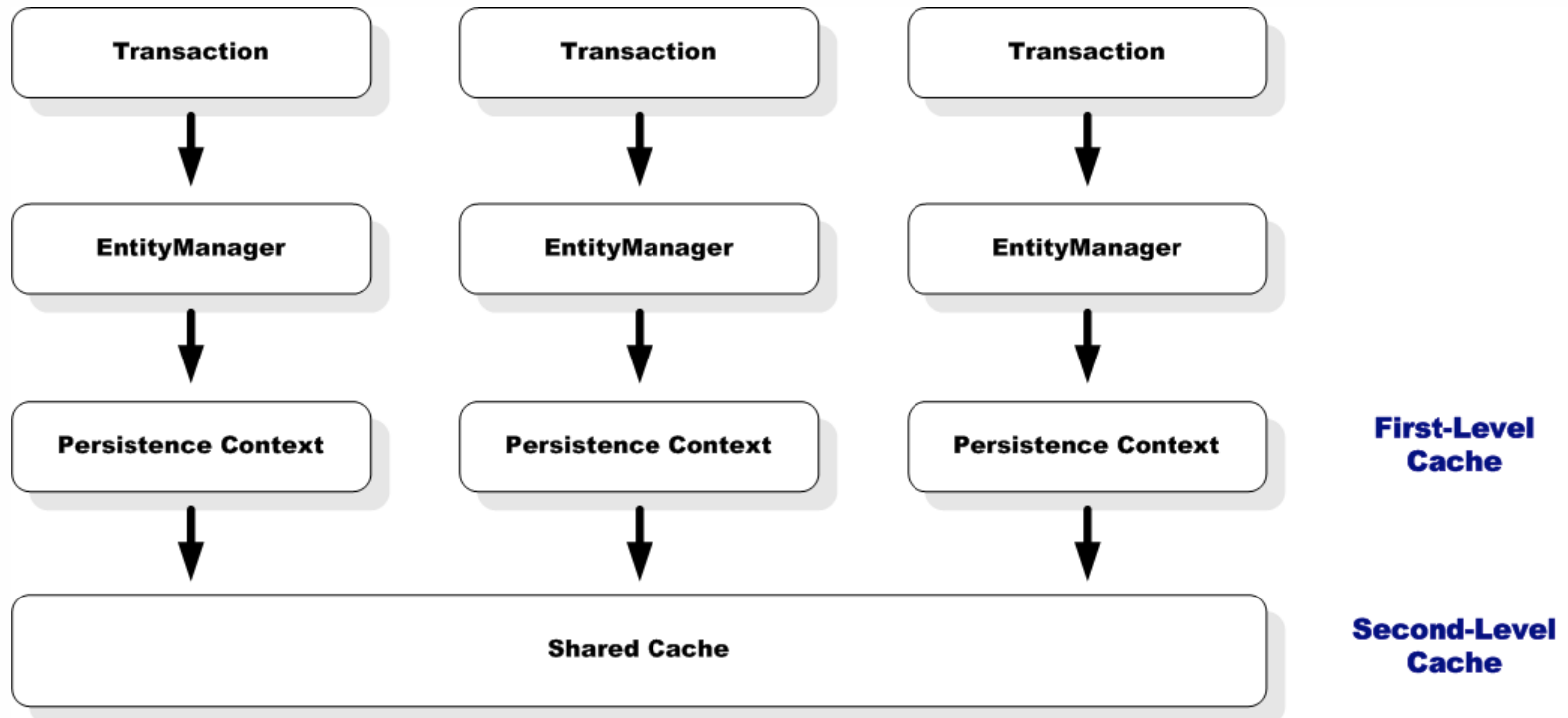


Domain/Infrastructure Tier

- **The persistence tier is an ideal point for caching since caching can be seen as a parallel concern to relational database centric persistence – particularly with ORM**
- **JPA 2 supports both transactional (1st level) and shared (2nd level) caching**
- **Persistence tier caching is well suited for domain objects with high read/write ratio**
- **Second level caching is largely an ORM adapter over existing distributed caches**



JPA 2 Caching





JPA Cache Configuration

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
  version="2.0">
  <persistence-unit name="AcmeBankDb">
    ...
    <shared-cache-mode>ENABLE_SELECTIVE</shared-cache-mode>
    <properties>
      ...
      <property name="hibernate.cache.provider_class"
        value="org.hibernate.cache.SingletonEhCacheProvider"/>
      <property name="hibernate.cache.use_second_level_cache"
        value="true"/>
      <property name="hibernate.cache.use_query_cache"
        value="true"/>
    </properties>
  </persistence-unit>
</persistence>
```



JPA Cache Configuration

```
@Entity
@Cacheable
@Cache(usage =
    CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
@Table(name = "CATEGORIES")
public class Category implements Serializable {
    @Id @GeneratedValue
    private Long id;
    private String name;
    @ManyToOne
    @JoinColumn(name = "PARENT_ID")
    private Category parent;
    @OneToMany(mappedBy = "parent")
    private Set<Category> subcategories;
    ...
}
```



JPA Cache Control API

```
public interface Cache {  
    public boolean contains(Class clazz, Object key);  
    public void evict(Class clazz, Object key);  
    public void evict(Class clazz);  
    public void evictAll();  
}
```



Resource (Database) Tier

- **Database/resource connection pooling essential caching technique for any serious production application**
- **Most application servers will also provide prepared statement caching as part of the connection pool**
- **Tune both connection pool size and prepared statement cache size**
- **You can also do statement caching on the database end
– talk to your DBA 😊**



Distributed Caches

- **Distributed caches are a key infrastructure piece for any caching or clustering solution**
- **They can be used directly in the application or persistence tier where they are needed (a relative rarity)**
- **It is a crowded field with many open source and commercial options**
- **Tools come in a variety of clustering, replication, transactionality, robustness and administrative features (cost and support are all over the map too)**



JCache (JSR-107) API Example

```
@Inject Cache cache;  
...  
cache.put(key, value);  
...  
String data = (String) cache.get("my-data");
```



Caching as an Aspect

```
public class DefaultBidDao implements BidDao {
    @Resource(name="jdbc/AcmeBankDB")
    private DataSource datasource;
    ...
    @Cache
    public Bid getBid(Long id) {
        ...
    }

    @FlushCache
    public void updateBid(Bid bid) {
        ...
    }
    ...
}
```



Java Distributed Caches

- **Coherence**
- **Terracotta**
- **GigaSpaces**
- **Infinispan/JBossCache**
- **EHCache**
- **JCS**
- **SwarmCache**
- **OSCache**



Summary

- **Caching an important optimization technique for vast majority of applications**
- **Each enterprise application tier is rife with opportunities for caching**
- **Some caching techniques are obvious and natural, others require more careful analysis**
- **Some are transparent while others are somewhat invasive**
- **Variety of options to match application needs**



References

- **Caching in HTTP,**
<http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html>
- **Apache Caching Module,**
http://httpd.apache.org/docs/2.0/mod/mod_expires.html
- **Resin Proxy Cache,**
<http://www.caucho.com/resin/admin/http-proxy-cache.xtp>
- **Oracle Coherence,**
<http://www.oracle.com/technetwork/middleware/coherence/overview/index.html>



References

- Terracotta, <http://www.terracotta.org>
- GigaSpaces, <http://www.gigaspaces.com>
- Infinispan/JBossCache, <http://www.jboss.org/infinispan>
- EHCACHE, <http://ehcache.org>
- JCS, <http://jakarta.apache.org/jcs/index.html>
- SwarmCache, <http://swarmcache.sourceforge.net>
- OSCache, <http://www.opensymphony.com/oscache/>



Thanks!

Email: reza@caucho.com

Twitter: cauchoresin

Facebook:

[http://www.facebook.com/pages/Caucho-Technology/
8671323790](http://www.facebook.com/pages/Caucho-Technology/8671323790)

Presentation materials: <http://www.caucho.com/articles/>