
BEA
world 2003
THE 8TH ANNUAL BEA TECHNOLOGY CONFERENCE
Tuning BEA WebLogic Portal
Austin Maher
Independent Consultant

Learning Objectives

BEA world 2003

- As a result of this presentation, you will be able to:
  - Plan a performance tuning effort
  - Increase the performance of your WebLogic Portal application.

bea

Speaker's Qualifications

- **Austin Maher** has been developing with object technology for 17 years. He is a certified BEA WebLogic Developer and an independent software consultant.
- **Austin Maher** was a senior principal consultant and the technical manager of the BEA RTP Accelerated Development Center from 2000 – 2002.

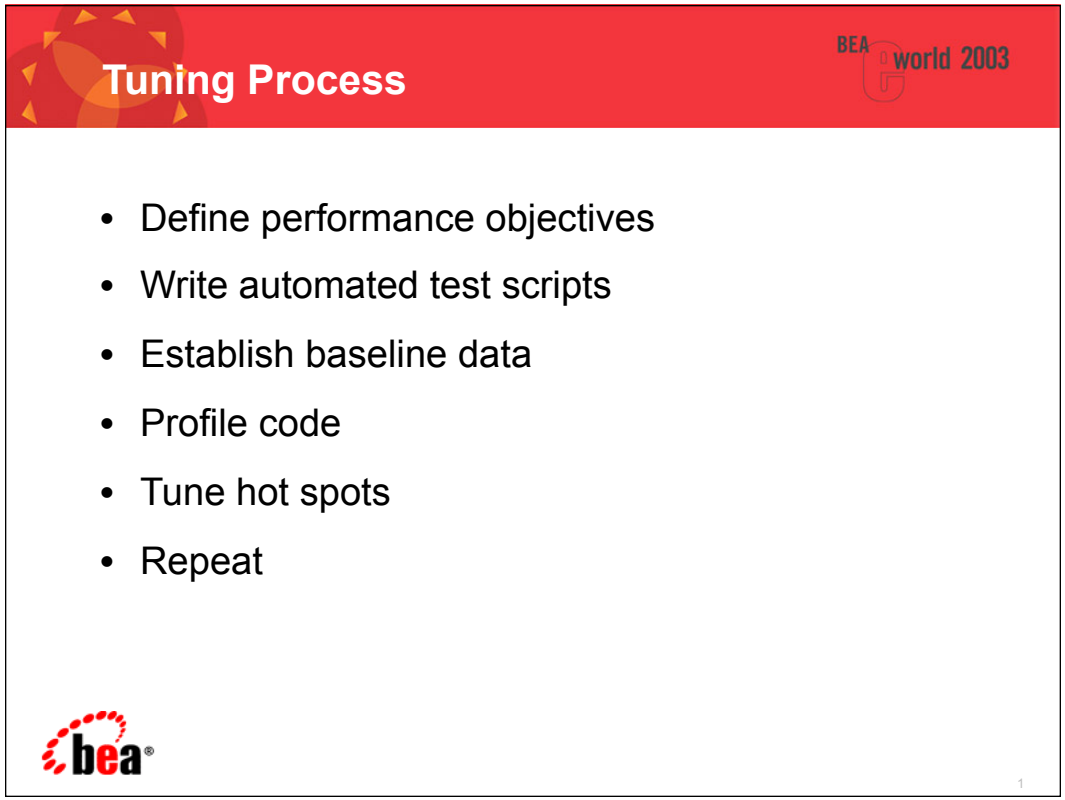
Presentation Agenda

BEA world 2003

- Preparing for tuning
- Finding out where the time goes
- Reducing execution time

bea

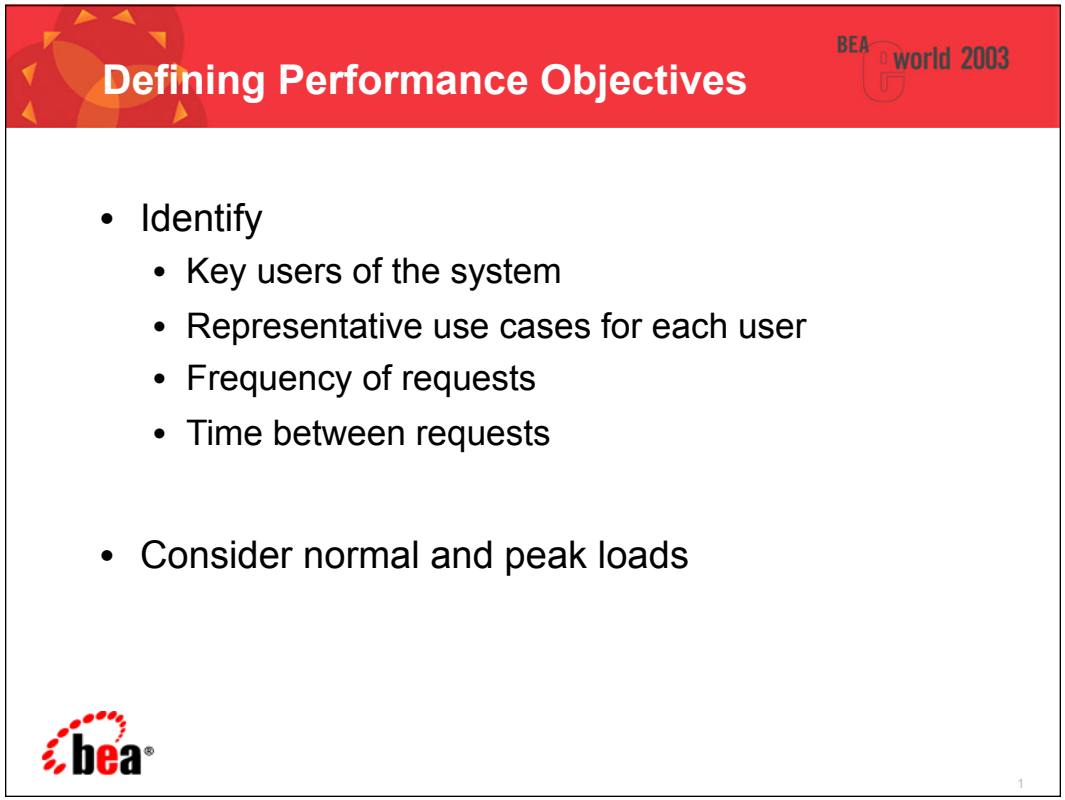
Preparing For Tuning



Projects can be pulled in conflicting directions when it comes to application tuning. One the one hand, project managers that identify performance as a potential development risk can try to organize system delivery in iterations, bringing in the performance-critical or performance-challenged components first. This allows the project team to measure and react to application performance early in the project delivery cycle.

On the other hand, there is the old adage: “Make it run, make it right, make it fast”. You don’t want to sacrifice application design and maintainability unless you know where the bottlenecks are (without a complete system, it’s hard to predict where you will get the biggest bang for your tuning buck).

Either of these sentiments may be correct, depending on the project. It’s always a good idea of explicitly manage project risk and prioritize the delivery of project deliverables in a manner that will reduce as much risk as possible. Questions that may help decide if early tuning is in order might include: how well can you predict what your site traffic patterns will be? How bad was the performance of that early performance snapshot? How well does the test environment reflect what the deployment environment will be?



Without objectives …

No way to determine adequate performance

No way to know when to stop optimizing

Contingency planning cannot take place until deployment

Confirming the feasibility of performance goals early in the project reduces risk and increases everyone's level of comfort

Performance objectives should be in terms of the customer's application, not abstract metrics. For example, 100 transactions per second is meaningless if you don't know what types of transactions you are measuring. Saying that 100 account reps can concurrently update customer profiles is precise and measurable.

If there's an existing system in place, use it to get benchmark data. This validates the system model and can be used to drive load tests or simulations.

Write Test Scripts

BEA world 2003

- A reusable test script facilitates a consistent, reproducible test environment.
- Build a test script for each key use case.

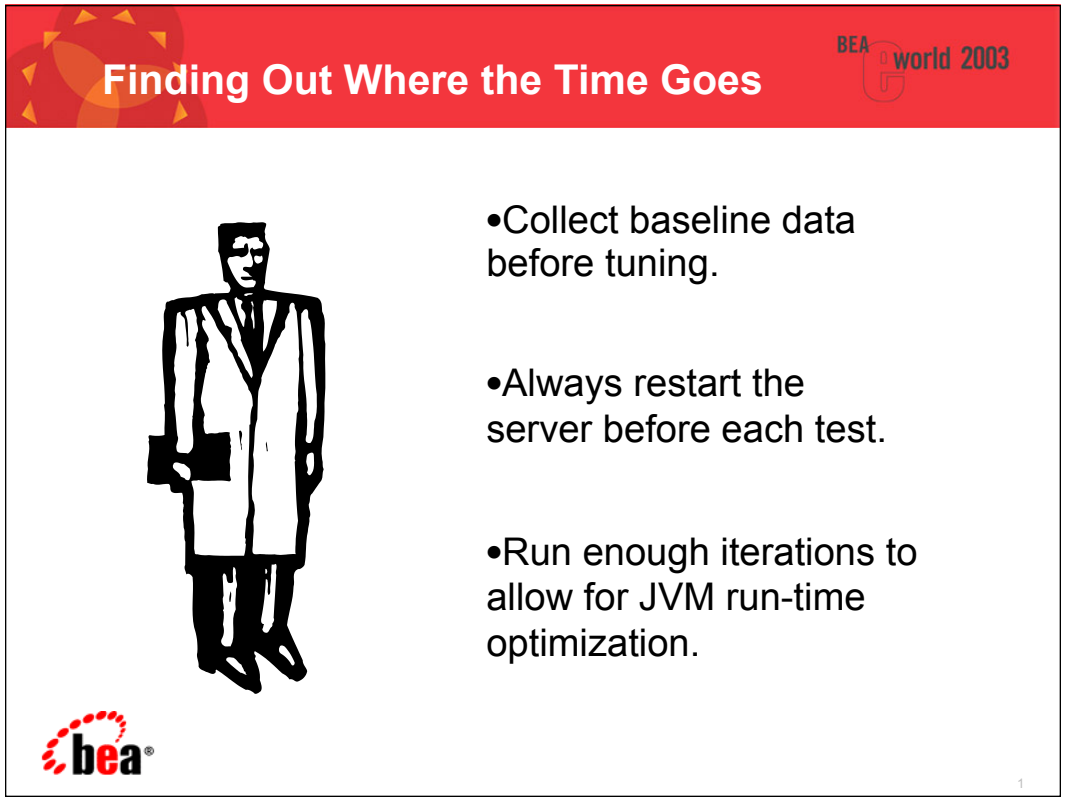
bea

1

Lots of scripting options. For example:
Mercury Interactive's LoadRunner
Grinder (http://grinder.sourceforge.net),
Apache JMeter,

Assemble Team

- Tuning requires people to monitor all the subsystems:
  - Web Server
  - Application Server - Operating System
  - Application Server - Java Virtual Machine
  - Database Server
  - Network



Collection data from

The operating system:

The Java Virtual Machine: (using WebLogic Console or SNMP)

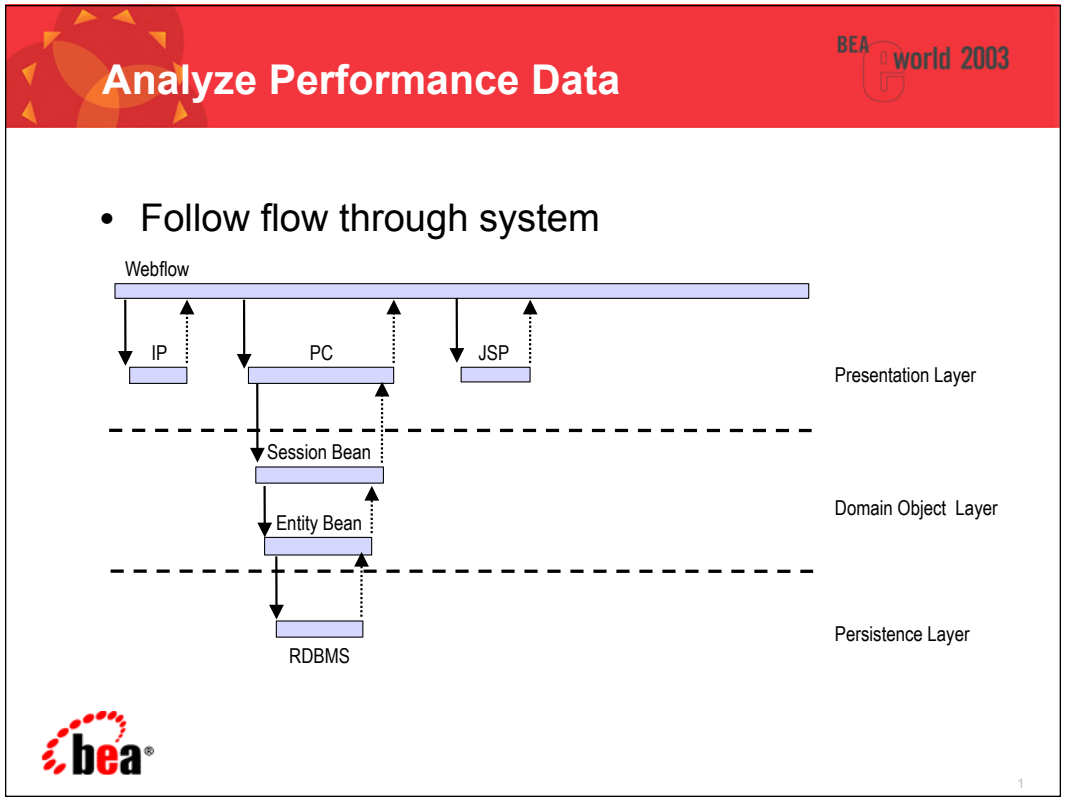
The Application: (using Log4J, Servlet Filters, Thread Dumps)

For an aggressive performance test, the WebLogic Console may measurably affect system capacity. In those cases, using SNMP to confirm system heartbeat may result in more accurate system capacity measurements.

Typical operating system monitoring software includes, Freshwater Software's® SiteScope®, Microsoft Windows Performance Monitor, top, Quest Software's Big Brother, IBM Tivoli

Java Virtual Machine monitors include: Sitraka JProbe, Wily Introscope, Altaworks Panorama.

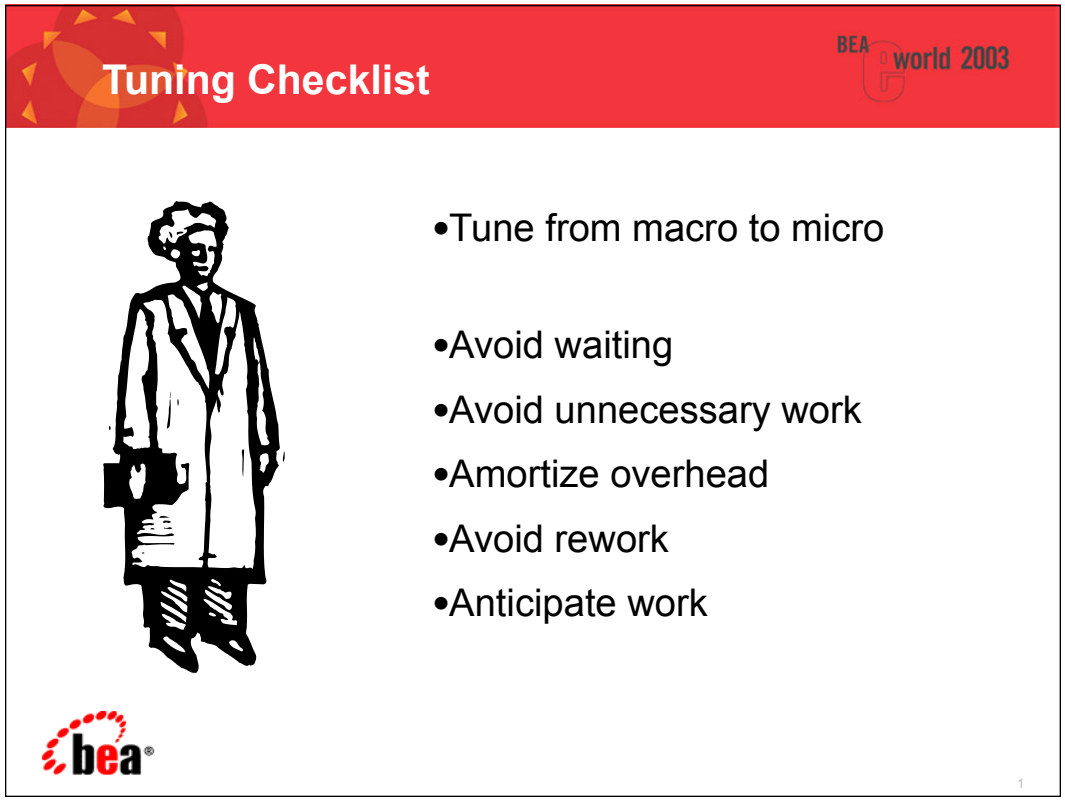
Note: for slow building or hung pages, periodic thread dumps can be used to troubleshoot where the software is waiting.



Since most caches allow an application to avoid calling from one layer of abstraction to another, manipulating a standard sequence diagram to isolate calls between the layers of the application makes the opportunities for caching clearer.

This notation can also be used to express where the time goes by drawing methods proportional to their execution time.

Reducing Execution Time



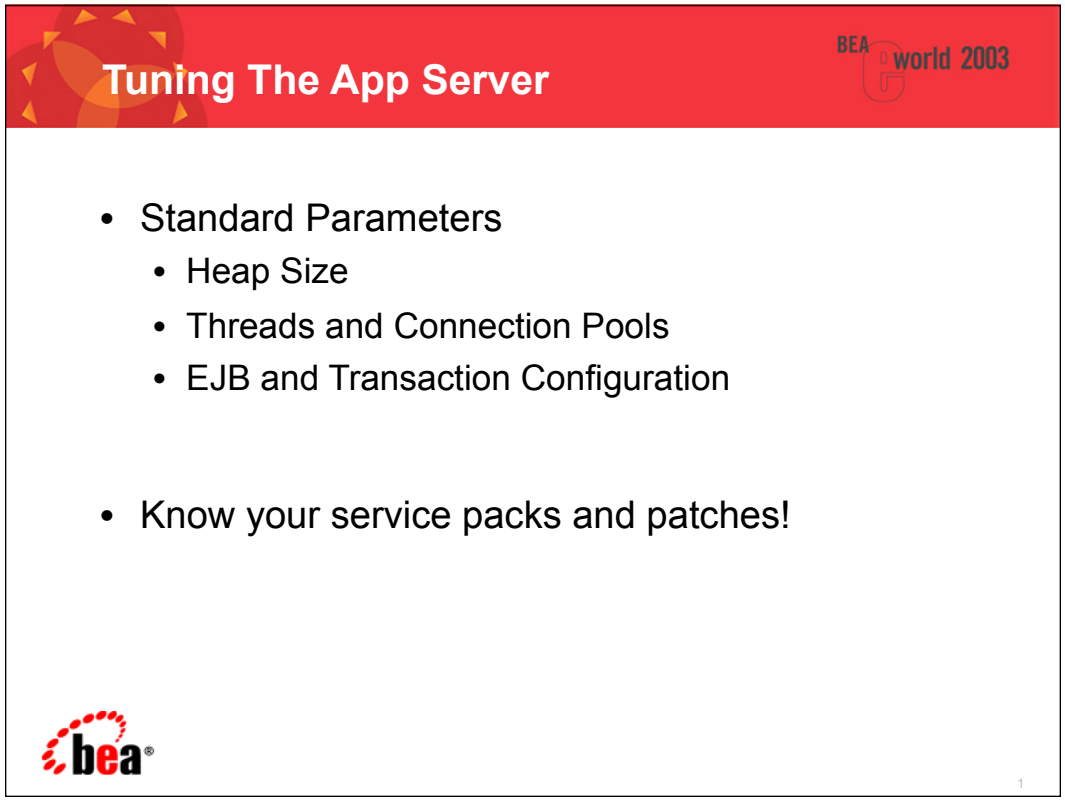
When it comes right down to it, Portal is just a J2EE Enterprise Application. The standard tuning advice for J2EE applications would hold true for Portal as well.

To avoid waiting, consider things like the number of threads in the app server, the number of database connections, other pool sizes (EJB and cache) and virtual memory size.

Amortizing overhead can be accomplished by making a smaller number of larger calls, for example when issuing database queries or defining database transactions.

To avoid rework, consider caching (at whatever level of the application architecture makes sense)

Anticipating work can be handled on application startup, for example using startup tasks or load on startup servlets.



Enabling JSP pre-compilation will increase startup, but allow for a more consistent user experience.

Production mode eliminates the overhead associated with dynamic discovery of applications.

There are also several EJB tuning considerations:

EJB Instance Pool

Max beans in pool

Initial beans in free pool

Max beans in cache

Transactions

Transaction boundaries

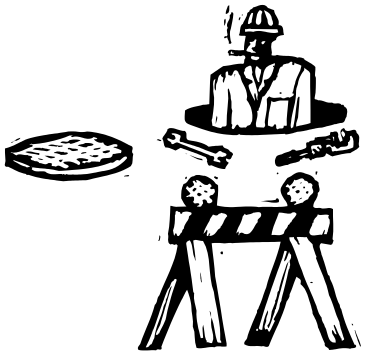
Concurrency strategy

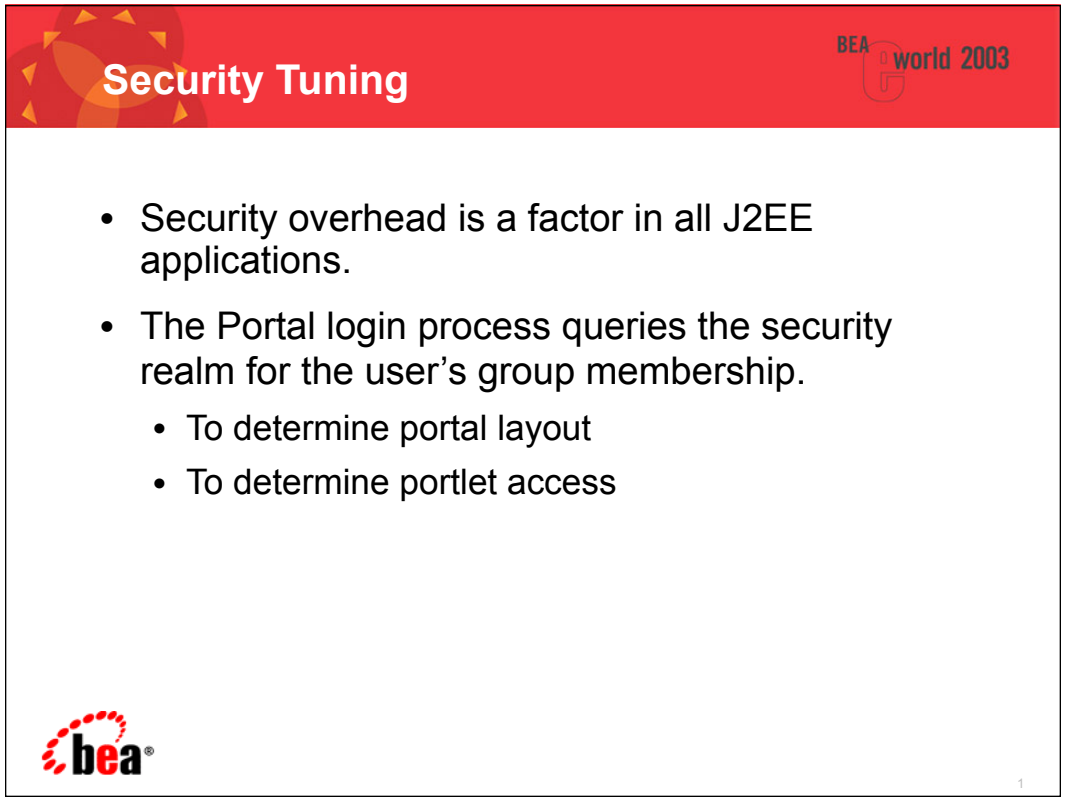
Transaction isolation level

Another potential optimization is to cache EJB Home lookups, avoiding the overhead of repeatedly generating the JNDI context and searching for the home object.

Portal Tuning Tips

- Security
- Caching
- Miscellaneous





There was a performance problem with RDBMSDelegate.getUser() and getGroupMembers() - CR 71557:
CR included in WLP 4.0 SP3 and WLP 7.0

Tuning Check for User In Group

- Portal iterates over all groups
  - calling group.isMember (Principal user)

- There are often a large number of groups, few of which are of interest to Portal.
- Iterating over lots of groups can take a lot of time.

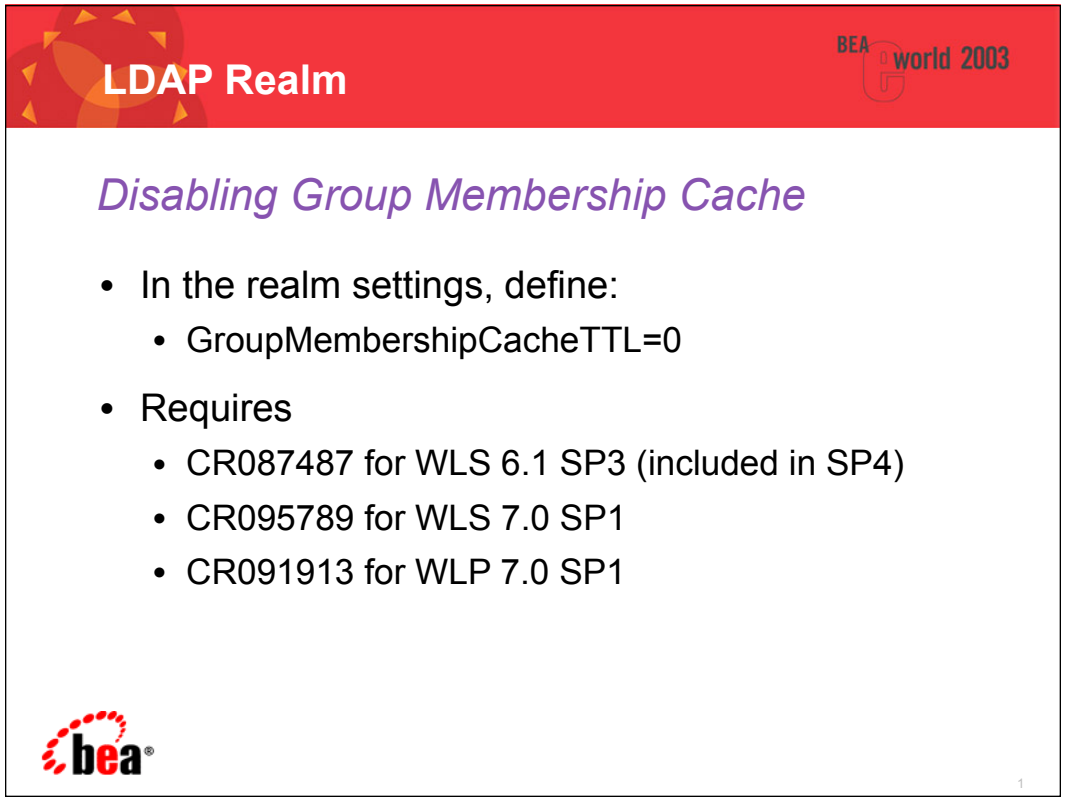
LDAP Realm

Tuning Check for User In Group

- Create a PortalGroup Organizational Unit under the Group Organizational Unit and give the Distinguished Name for that OU to the LDAP Security Realm as the Group Base DN.

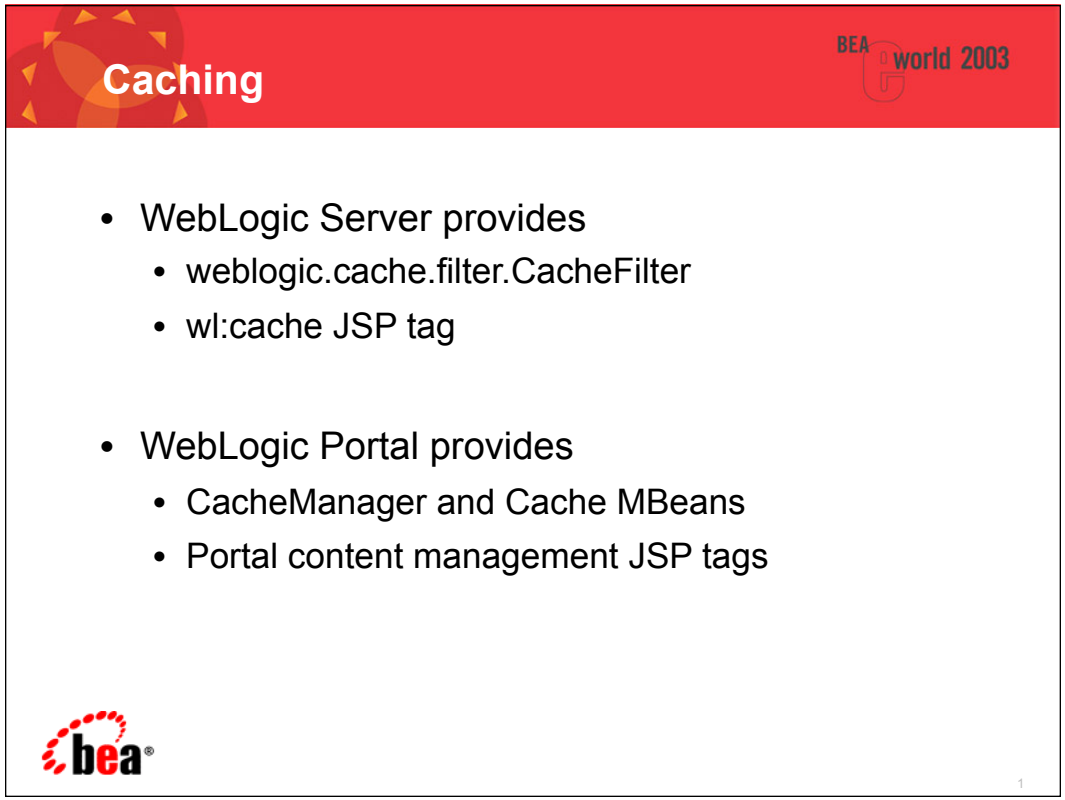
Disabling Group Membership Cache

- Groups can cache their members.
- Reading in all the members of a group can be prohibitively expensive .



The CR095789 rolled a bunch of WLS changes into the weblogic.security.ldaprealmv2.LDAPRealm (WLS code).

The CR091913 patch is just portal code. It is the com.bea.p13n.security.realm.PortalLDAPRealm (Portal code) that extends the new weblogic.security.ldaprealmv2.LDAPRealm.



The primary way to avoid unnecessary work is by judiciously caching whatever data can be cached.

Of course to determine what can be cached, you have to

Understand data update frequencies

Evaluate the cost of accessing the data

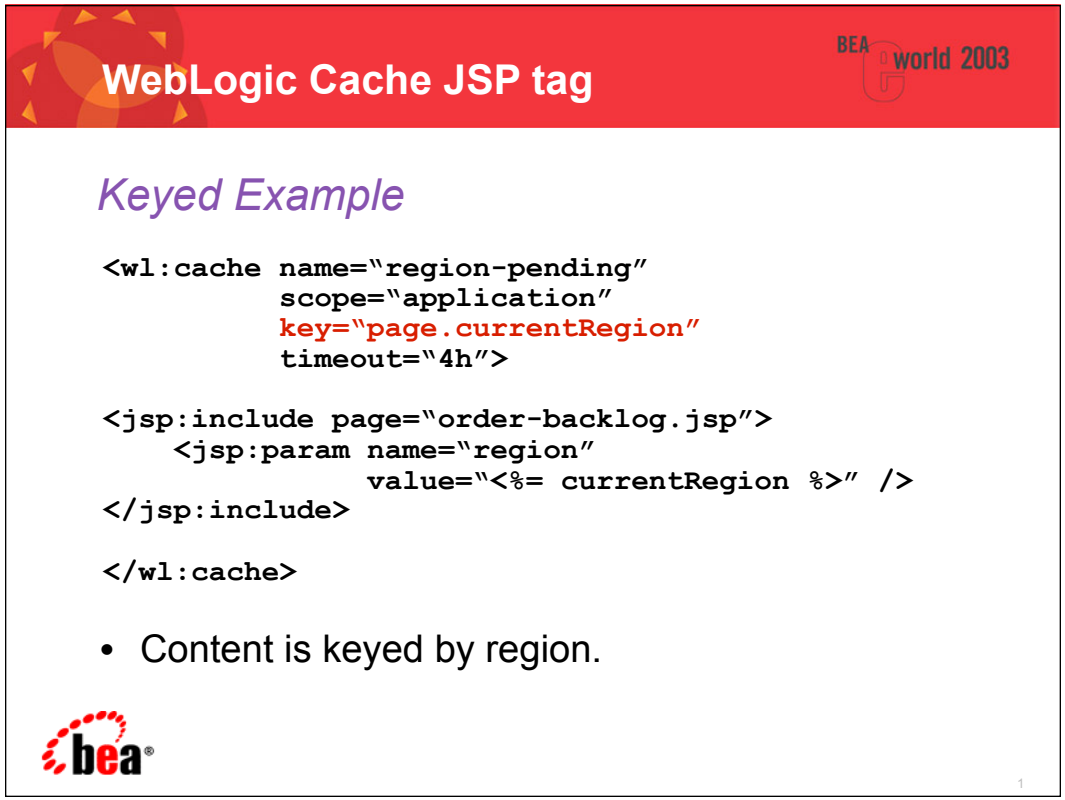
Evaluate the cost of showing obsolete/invalid data.

WebLogic Caching

- Handy for caching
  - Pages
  - Portlets
  - HTML fragments
- Data saved at any scope:
  - Application, Session, Page, Request

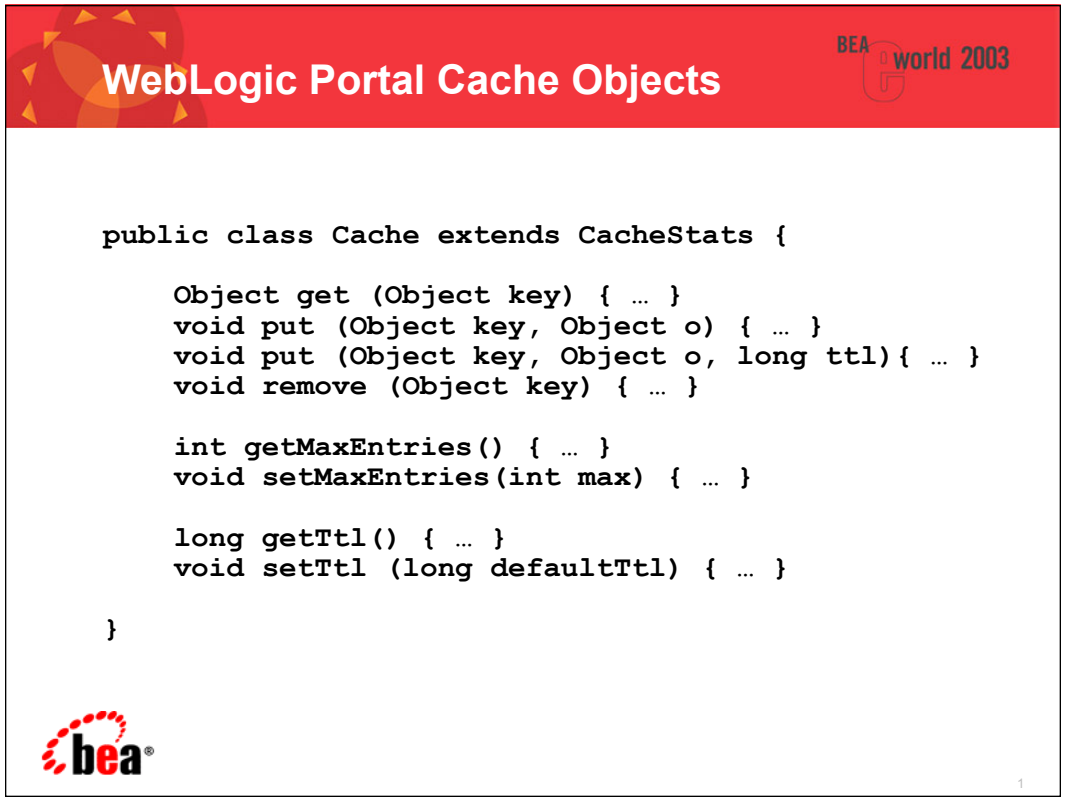
Note: WebLogic Server provides both JSP tags and servlet filters that cache – with equivalent functionality. The following slides only cover the JSP tags, but the same functionality/options are accessible as servlet filters by configuring the servlet filters in your web.xml file.

Caching the results of a jsp:forward will not work. The generated Java code forces a return after the forward call (so the wl:cache clock is never closed).



Cached Output Data Example

The request attribute “currentBacklog” can be accessed beyond the cache block



Handy for caching application and user data

Used heavily by the Portal framework

WebLogic Portal Cache Factory

```
public class CacheFactory {

     static boolean cacheExists (String name) { … }
     static Cache getCache (String name) { … }
     static String[] getCacheNames () { … }
     static void removeCache (String name) { … }

}
```

- CacheFactory provides simple programmatic interface to object caches.

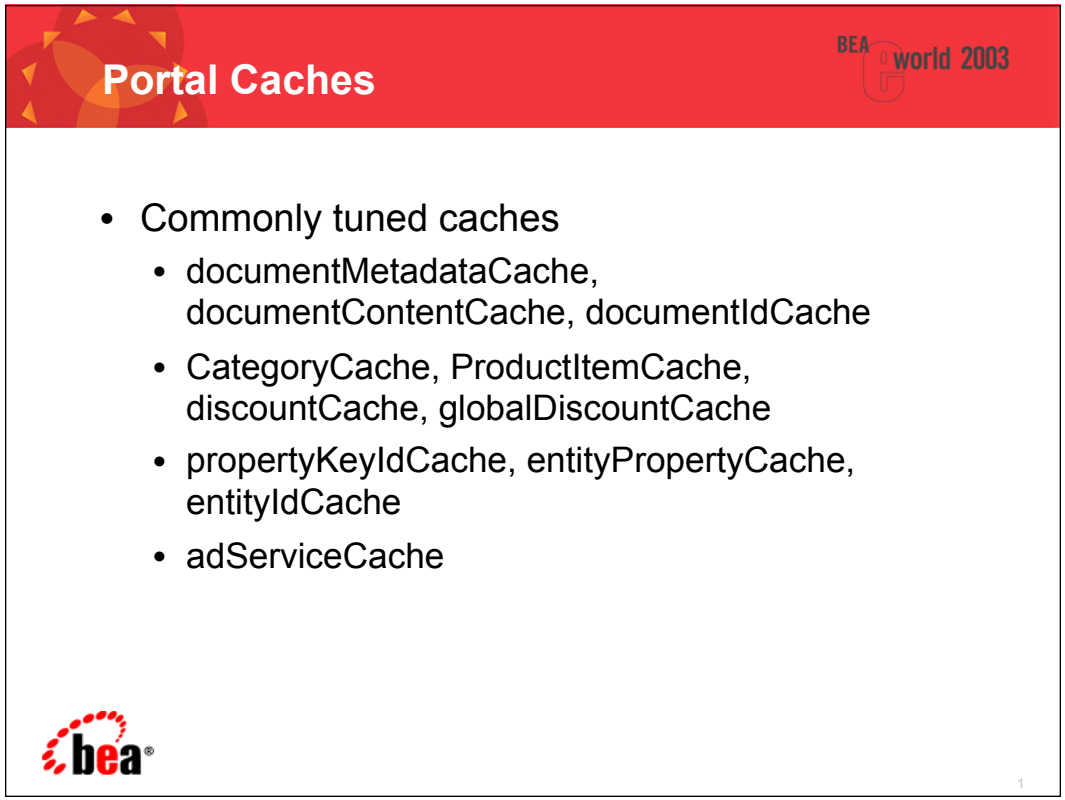
The most common caches are defined in application-config.xml

There are other standard caches that are only accessible programmatically.

Caches specified in application-config.xml can be configured from the WebLogic Console.

You can add existing prgrammatically accessed caches to application-config.xml

All caches can be accessed via the CacheFactory or the CacheManager MBean.



More information on Portal caches is available using the JSP found at dev2dev:

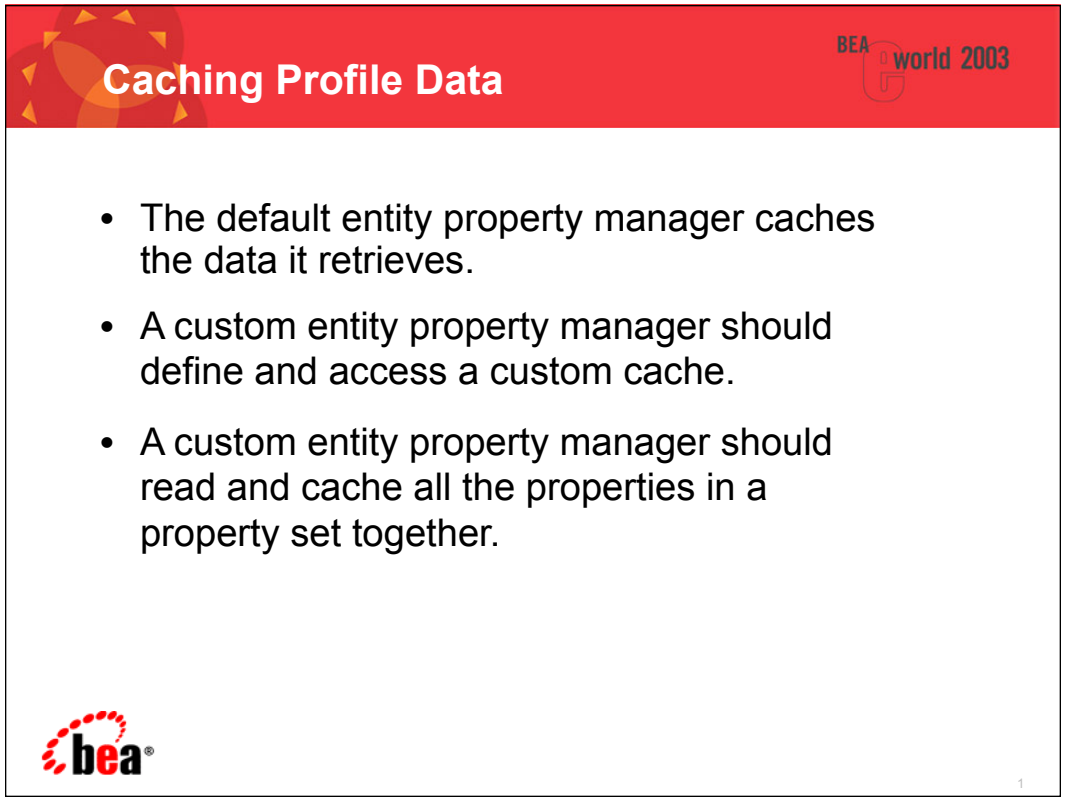
http://dev2dev/code/codedetailcontent.jsp?productType=weblogic+portal&codeType=code+sample&filepath=components%2Fdev2dev%2Fcodelibrary%2Fcodesamples%2FWLP_Cache_List_JSP.htm

Caching and Clustering

- Caches are not replicated.
- Caches listed in application-config.xml can be configured and cleared via the console.

Applications of Caching

- Profile Data
- Content Queries
- Portlet Refresh Events
- Catalog and Discount Data



The link for a sample user profile on dev2dev:

http://dev2dev.bea.com/code/codedetailcontent.jsp?productType=weblogic+server+7.0&codeType=code+sample&filepath=components%2Fdev2dev%2Fcodelibrary%2Fcodesamples%2Fcodesample_wlp_uup.htm

The source can be downloaded from:

ftp://edownload:BUY_ME@ftpna2.bea.com/pub/downloads/UUPExample_WLP70.zip

Caching Content Query Results

- The Portal JSP tags include support for caching the results of content queries:
  - cm:select
  - cm:selectById
  - pz:contentQuery

Caching Content Query Results

Caching at Application Level

```
<pz:contentQuery
    query="DOC_TYPE = 'news-body'"
    sortBy="creationDate"
    contentHome="<%= DOCUMENT_MGR_HOME %>"
    id="newsDocs"
    useCache="true"
    cacheId="general-news"
    cacheTimeout="86400000"
    cacheScope="application" />
```

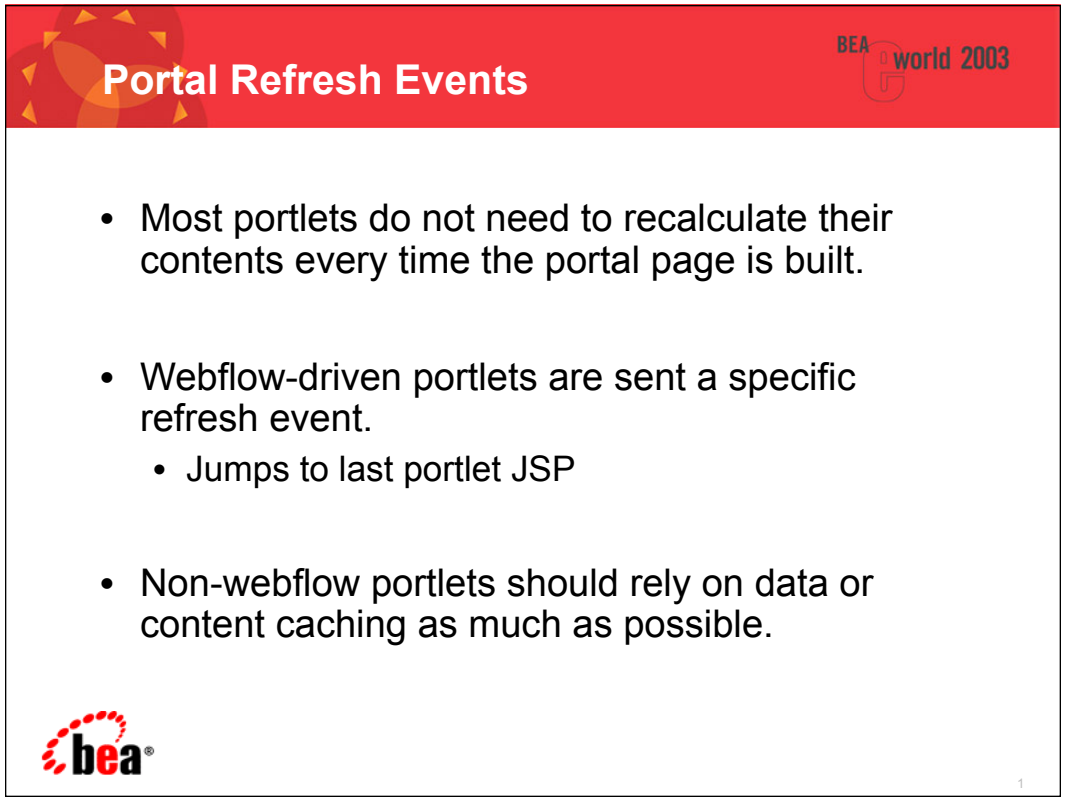
- Show the same news documents to unrecognized visitors for 24 hours.

Caching Content Query Results

Caching Visitor-Specific Results

```
<pz:contentQuery
    query="<%= userNewsInterests %>"
    sortBy="creationDate"
    contentHome="<%= DOCUMENT_MGR_HOME %>"
    id="newsDocs"
    useCache="true"
    cacheId="personal-news"
    cacheTimeout="1800000"
    cacheScope="session" />
```

- Re-query news of interest to a logged in visitor every 30 minutes.



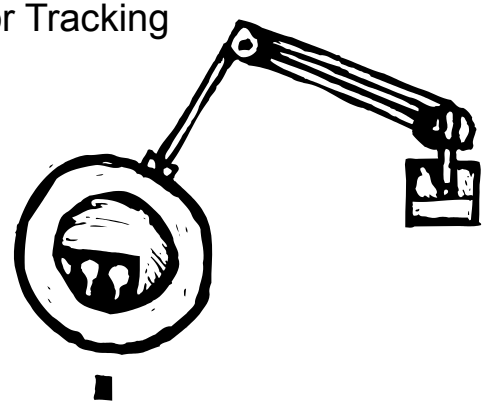
Using the standard lastContentURL action can be problematic when there's cross-portlet communication or other situations where a portlet other than the one with the focus needs to change the way it gets displayed.

Catalog and Discount Data

- Ideally the category and product item caches would be large enough to hold the entire catalog.

Other Tuning Tips

- Caching the Home Page
- Campaign Scenario Actions
- Events and Behavior Tracking
- Ad Display Buffer



Caching the Home Page

- All unrecognized visitors typically see the same portal home page.

```
<% if (request.getRemoteUser() == null) { %>

<jsp:forward page="staticHome.html" />

<%
} else {
%>

<jsp:forward page="application" />

<% } %>
```

Campaign Action Definition

- Campaign actions are organized by scenario.
- Scenarios are tied to a specific segment.

- Choosing narrower segments reduces the number of rules that get evaluated.

Campaign Action Definition

BEA world 2003

Example

- GoldScenario with 5 ad placeholder rules each for:
  under30, 30to60 and over60

  vs

  GoldUnder30Scenario, Gold30to60Scenario and GoldOver60Scenario:
  each with 5 ad placeholder rules

Events and Behavior Tracking

BEA world 2003

- Disable the BehaviorTrackingListener if you are not writing events.
- Tune behavior tracking event pool.
  - Max event buffer size
  - Max time until write
  - Poll frequency

Behavior Tracking settings are defined in application-config.xml

Ad Display Buffer

BEA world 2003

- Campaigns can build rules that limit the number of times a piece of content is displayed.
- The number of times each piece of content is displayed in a placeholder is cached and periodically flushed to the database.
- Update the AdService parameter DisplayFlushSize.

bea

1

You can set the AdService's DisplayFlushSize parameter in application-config.xml.

Summary

- Establish performance objectives.
- Iteratively tune from macro to micro.
- Eliminate waiting by tuning the app server.
- Eliminate unnecessary rework by caching.

Acknowledgements

- Invaluable assistance was provided by:
  - Ture Hoefner
  - Peter Laird
  - Doug Leeper
  - Ron Schweikert
  - Sherwood Zern


BEA
world 2003
THE 8TH ANNUAL BEA TECHNOLOGY CONFERENCE
Q&A


BEA
world 2003
THE 8TH ANNUAL BEA TECHNOLOGY CONFERENCE
Thank you!


BEA
world 2003
THE 8TH ANNUAL BEA TECHNOLOGY CONFERENCE