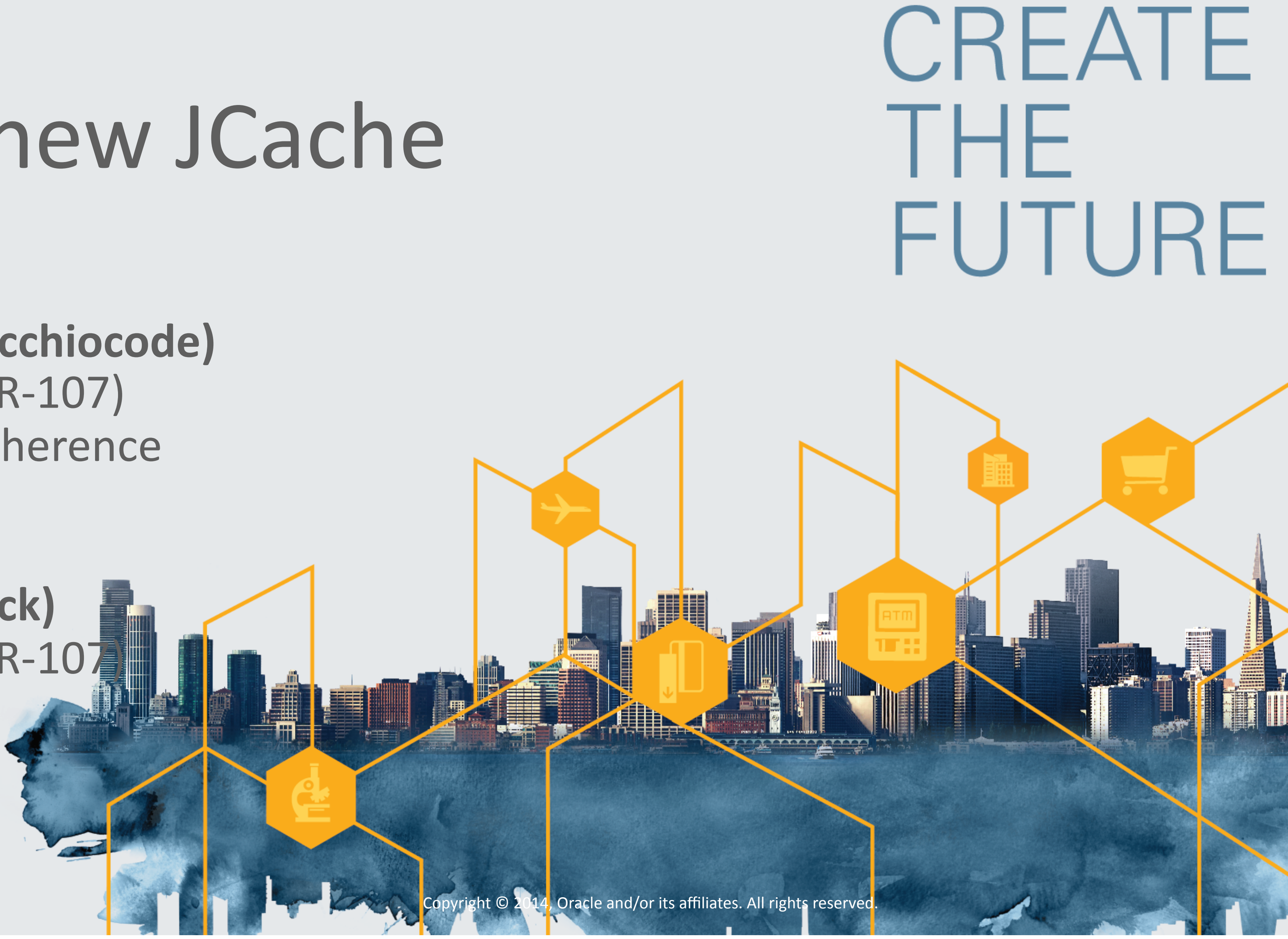
JavaOne™

ORACLE®

sing the new JCache

n Oliver (@pinocchiocode)
he Spec Lead (JSR-107)
itect | Oracle Coherence
cle Corporation

g Luck (@gregrluck)
he Spec Lead (JSR-107)
| Hazelcast

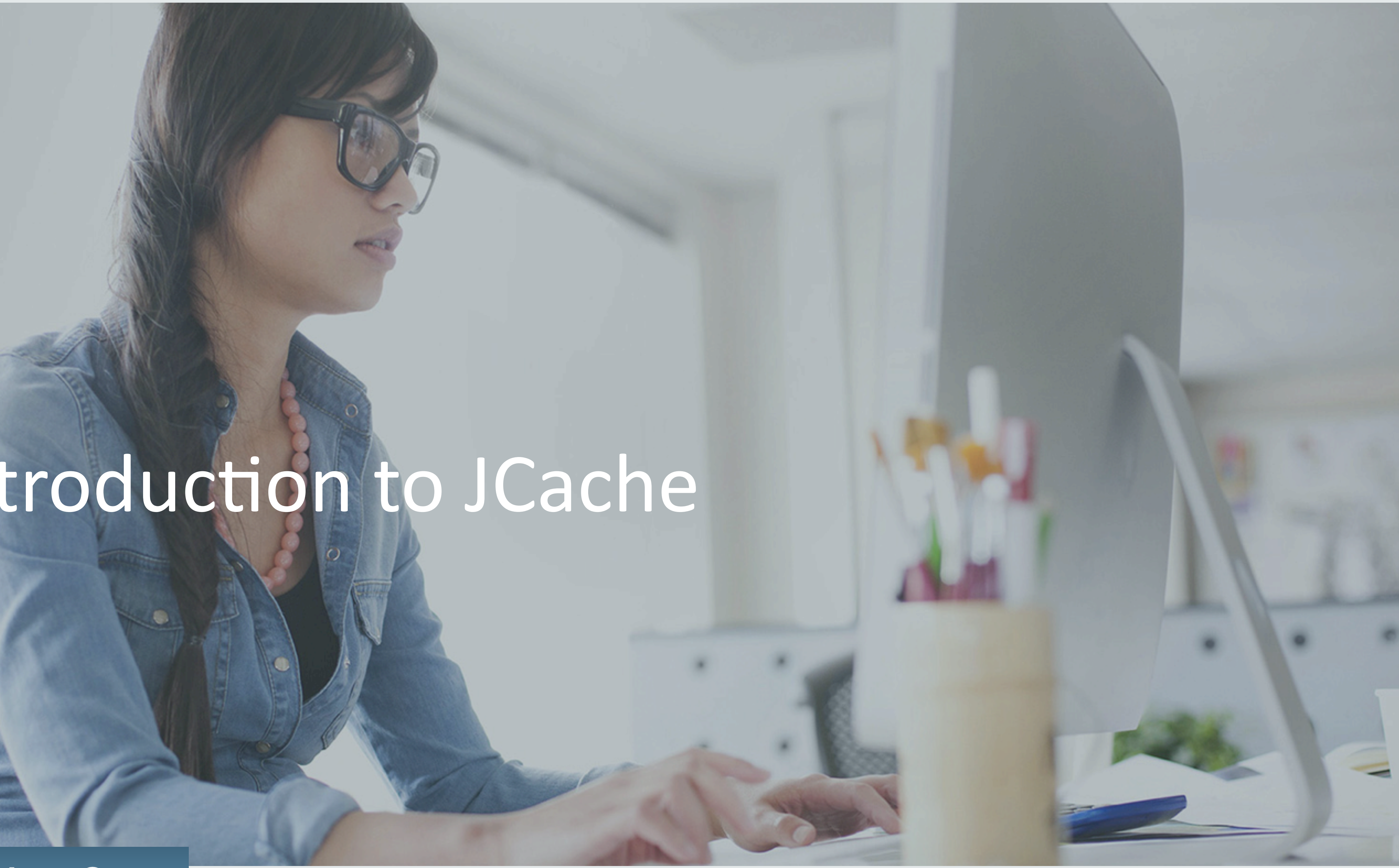


e Harbor Statement

following is intended to outline our general product direction. It is intended for
rmation purposes only, and may not be incorporated into any contract. It is not a
mitment to deliver any material, code, or functionality, and should not be relied u
aking purchasing decisions. The development, release, and timing of any features
tionality described for Oracle's products remains at the sole discretion of Oracle.

enda

JavaOne™ ORACLE



troduction to JCache

JavaOne™ ORACLE

enda

roduction to JCache

ache == Caching for the Java Platform

Produced via JSR-107

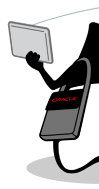
Ratified March 2014

Over 10 years of “Incubation”

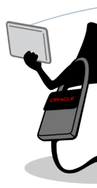
otivation

Standardize Caching Concepts, Terminology and API

Provide a mechanism for application portability



roduction to JCache



ommunity Driven

Leadership: Greg Luck, Brian Oliver

Expert Group: 10+ Companies, 8+ Individuals

rget Platforms

JCache Deliverable	Platform
Specification (SPEC)	Java 6
Reference Implementation (RI)	Java 7
Technology Compatibility Kit (TCK)	Java 7
Examples / Demos	Java 7

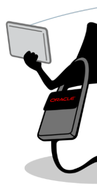
roduction to JCache

hich do you need?

java.util.Map (Java 6/7)
Key-Value Based API
Supports Atomic Updates
Entries Don't Expire
Entries Aren't Evicted
Entries Stored On-Heap
Store-By-Reference

javax.cache.Cache (Java 6)
Key-Value Based API
Supports Atomic Updates
Entries May Expire
Entries May Be Evicted
Entries Stored Anywhere (ie: topologies)
Store-By-Value and Store-By-Reference
Supports Integration (ie: Loaders / Writers)
Supports Observation (ie: Listeners)
Entry Processors
Statistics

roduction to JCache



oject

JCP Project:
- http://jcp.org/en/jsr/detail?id=107

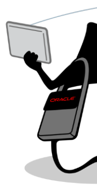
Source Code:
- https://github.com/jsr107

Forum:
- https://groups.google.com/forum/?fromgroups#!forum/jsr107

roduction to JCache

ven Dependency Information (Maven Central)

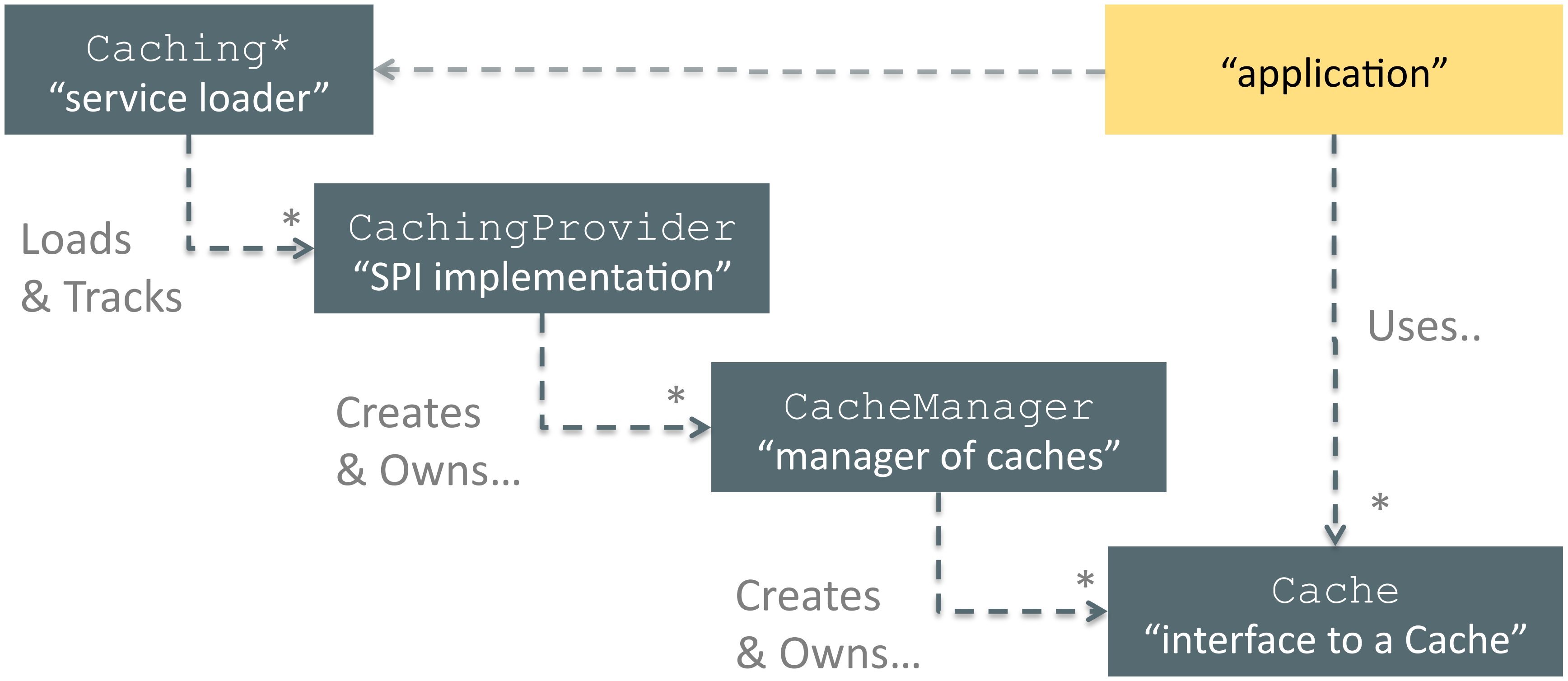
```
!– JCache Specification -->
dependency>
  <groupId>javax.cache</groupId>
  <artifactId>cache-api</artifactId>
  <version>1.0</version>
/dependency>
```



etting Started

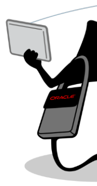
enda

runtime…


Caching*
“service loader”
“application”
Loads
& Tracks
*
CachingProvider
“SPI implementation”
Uses..
Creates
& Owns…
*
CacheManager
“manager of caches”
*
Creates
& Owns…
*
Cache
“interface to a Cache”

tting Started

- nplementations
  - *JCache Reference Implementation*
  - Oracle Coherence
  - Terracotta Ehcache
  - Hazelcast
- eep Track
  - https://jcp.org/aboutJava/communityprocess/implementations/jsr107/index.htm



sing Caches

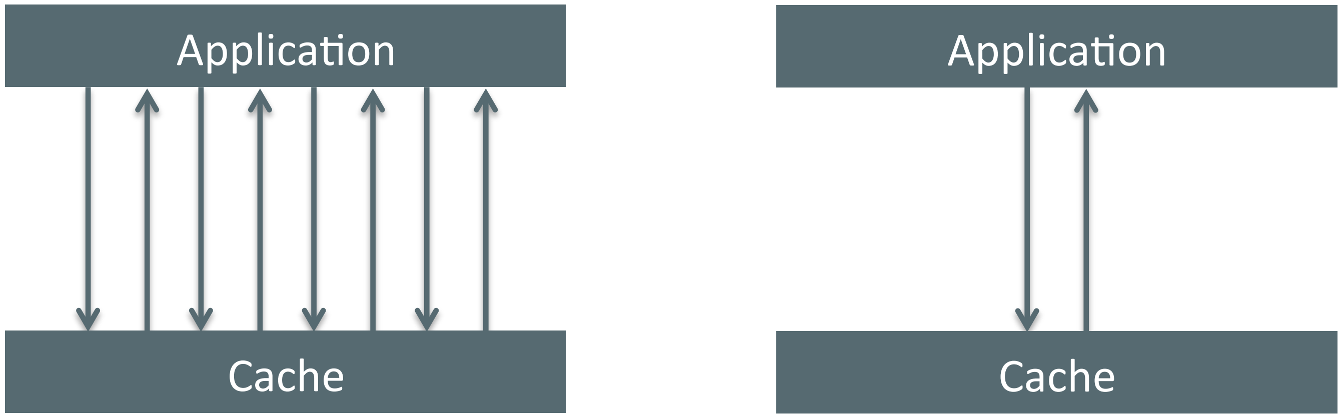
enda

try Processors

enda

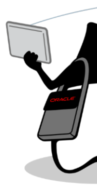
try Processors

- ustomizable Atomic Operations
- liminate Round-Trips! (in distributed systems)



- nable development of a Lock-Free API! (simplifies applications)
- May need to be Serializable (in distributed systems)

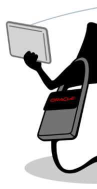
try Processors



```
 using an entry processor?
t value = cache.invoke(
                  "key",
                  new IncrementProcessor<>(), 42);


 using a lock based API? (which doesn't exist)
che.lock("key");
t current = cache.get("key");
che.put("key", current + 42);
che.unlock("key");
```

try Processors



- va 8 ready!
  - Use Lambdas as Entry Processors!
  - May need to be careful about serialization in distributed implementations

steners

enda

nnotations

enda

JavaOne™ ORACLE

notations

ache defines standard Caching annotations cover the most common
che operations:

`CacheResult`

`CachePut`

`CacheRemove`

`CacheRemoveAll`

ly Annotated Class Example

```
heDefaults(cacheName = "blogManager")
ic class BlogManager {
@CacheResult
public Blog getBlogEntry(String title) {...}

@CacheRemove
public void removeBlogEntry(String title) {...}

@CacheRemoveAll
public void removeAllBlogs() {...}

@CachePut
public void createEntry(@CacheKey String title, @CacheValue Blog blog) {...}

@CacheResult
public Blog getEntryCached(String randomArg, @CacheKey String title){...}
```

e Future?

enda

e Future?

ache 1.1 (2015)

Possible Maintenance Release? (helper classes to make it easier)

ache 2.0 (2015-2016)

Java 8 Language Features (Lambda & Streams)

Servlet 4.0 Integration / Session Caching?

Java EE 8 Alignment?

ache 3.0 (2017?)

Java 10 Language Features?

e Harbor Statement

preceding is intended to outline our general product direction. It is intended for
rmation purposes only, and may not be incorporated into any contract. It is not a
mitment to deliver any material, code, or functionality, and should not be relied u
aking purchasing decisions. The development, release, and timing of any features
tionality described for Oracle's products remains at the sole discretion of Oracle.

uestions?

rian Oliver (@pinocchiocode)

reg Luck (@gregrluck)

JavaOne™

ORACLE®