Designing Manageable Java™ Platform, Enterprise Edition (Java EE Platform) Applications in a Clustered Environment

Peter Kristiansson
Senior Specialist
Ericsson

Jens Jensen
Expert
Ericsson

BOF-4945

ERICSSON
TAKING YOU FORWARD

Java

Sun microsystems

Peter Kristiansson
Senior Specialist
Ericsson

Jens Jensen
Expert
Ericsson

Learn how to implement configuration support
- for clustered Java EE applications
- with advanced requirements
- in an easy way

GOAL

Agenda

- The Requirements
- A JMX Solution
- Code Samples
- Summary

Configuration Requirements

- Standardized API
- Changes at runtime
  - atomic changes
  - change notifications
- Object hierarchies
- Object validation
- Meta data available offline
- Efficient read access

Agenda

- The Requirements
- A JMX Solution
- Code Samples
- Summary

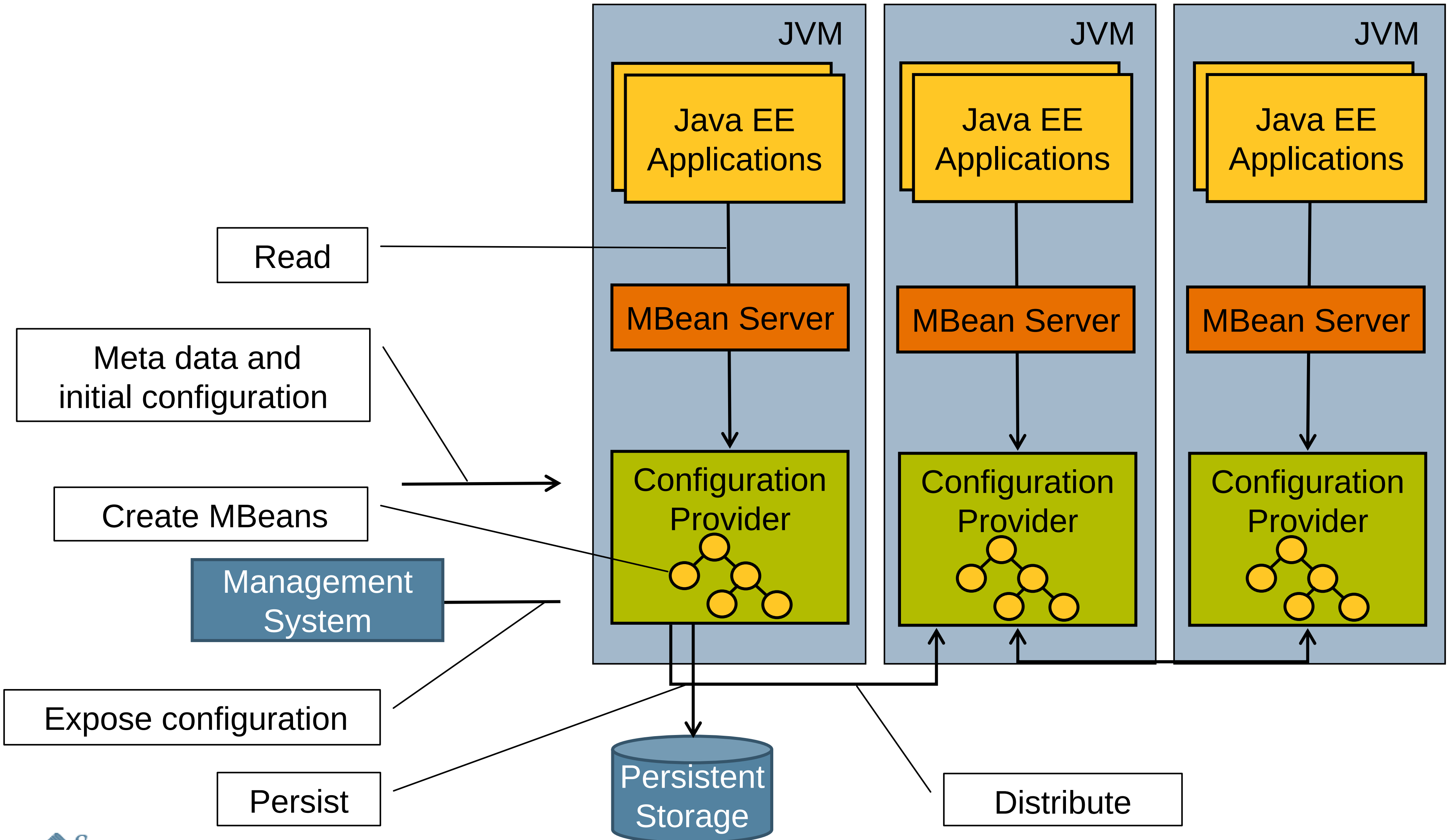
JMX Traditional Usage

- Through JMX the applications
  - implement MBeans
  - expose configuration and meta data to management systems
  - validate configuration changes
- Through other APIs the applications
  - read configuration from a persistent storage
  - write configuration to the persistent storage

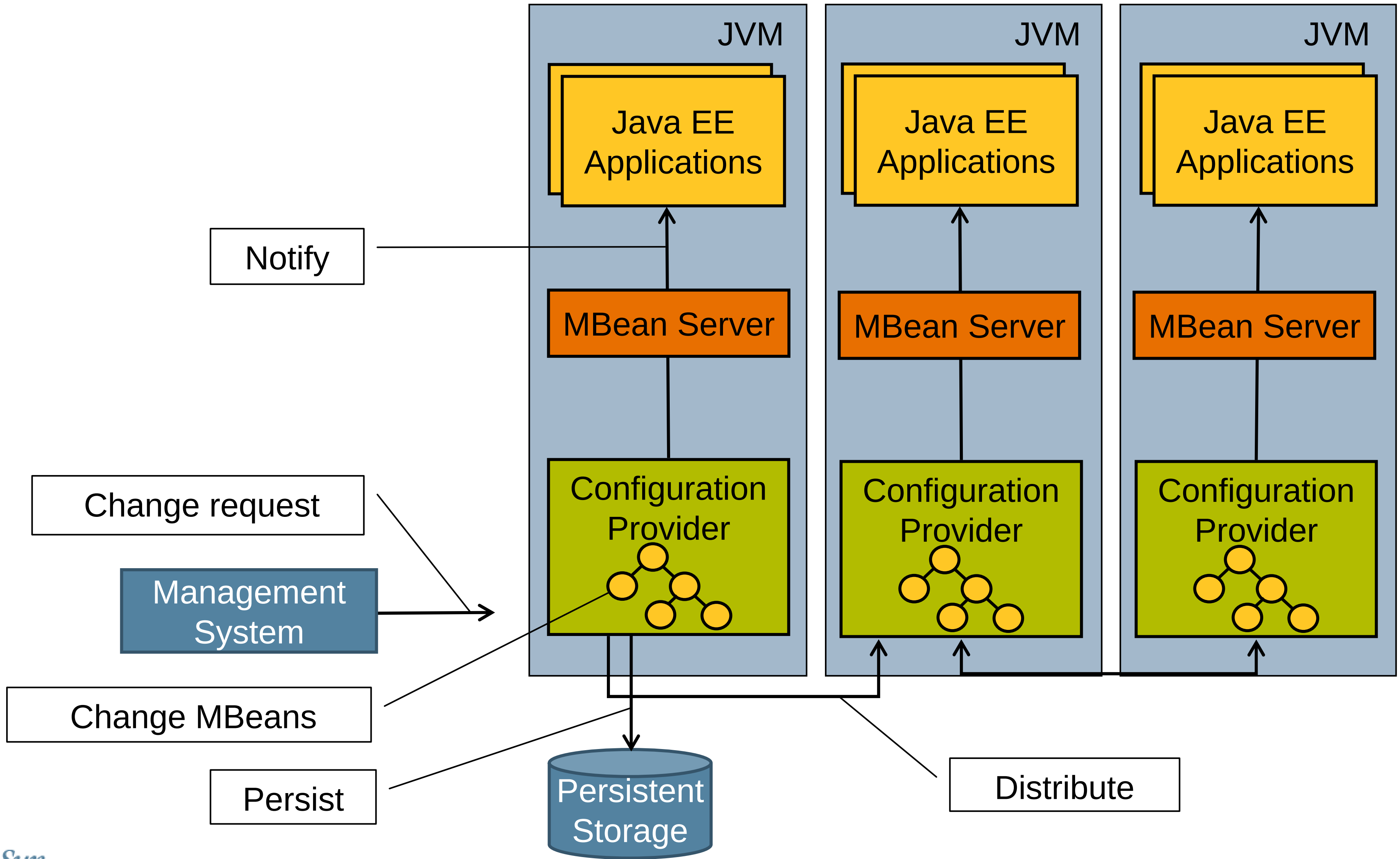
JMX Alternative Usage

- The applications
  - do not implement any MBeans
  - provide configuration meta data in XML
  - do not handle persistence
  - read the configuration through JMX
- A configuration provider
  - creates MBeans based on meta data
  - exposes configuration to applications and management systems
  - validates changes
  - handles persistence and distribution
  - is generic

Configuration at Application Deployment



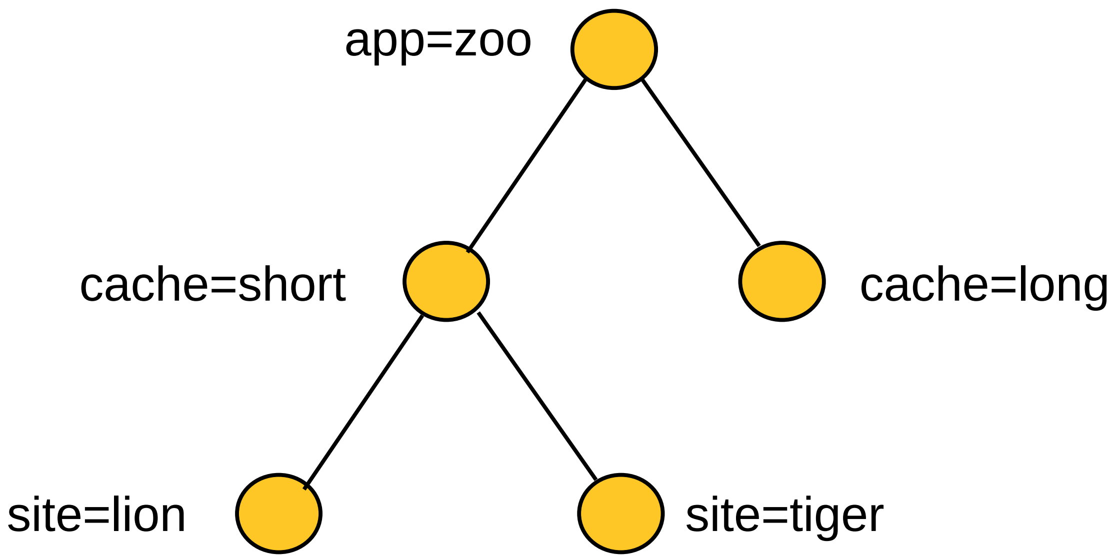
Configuration Changes at Runtime



Atomic Configuration Changes

- The applications
  - read all attributes in one operation
  - receive all changed attributes in one notification
- The configuration provider
  - publishes external interface allowing bundled changes
  - synchronizes changes and read operations

Configuration Object Hierarchies



Java

Configuration Object Hierarchies

- JMX object names: flat structure

Domain name

Unordered set of...

$$\underbrace{\overbrace{\texttt{DefaultDomain}}^{\text{Domain name}}:\overbrace{\underbrace{\texttt{type=printer}}_{\text{...key value pair}},\texttt{name=HPLaserJet4250\_room17}}^{\text{Unordered set of...}}}$$

```
DefaultDomain:type=printer,name=HPLaserJet4250_room17
```

...key value pair

- Distinguished names: hierarchical structure

Ordered sequence (right to left) of...

```
site=lion,cache=short,app=zoo
```

...key value pair
with the name of one object in the tree

Configuration Object Hierarchies

- Combining JMX object names with distinguished names

Domain name

A key value pair with key DN and a value of a sequence of...

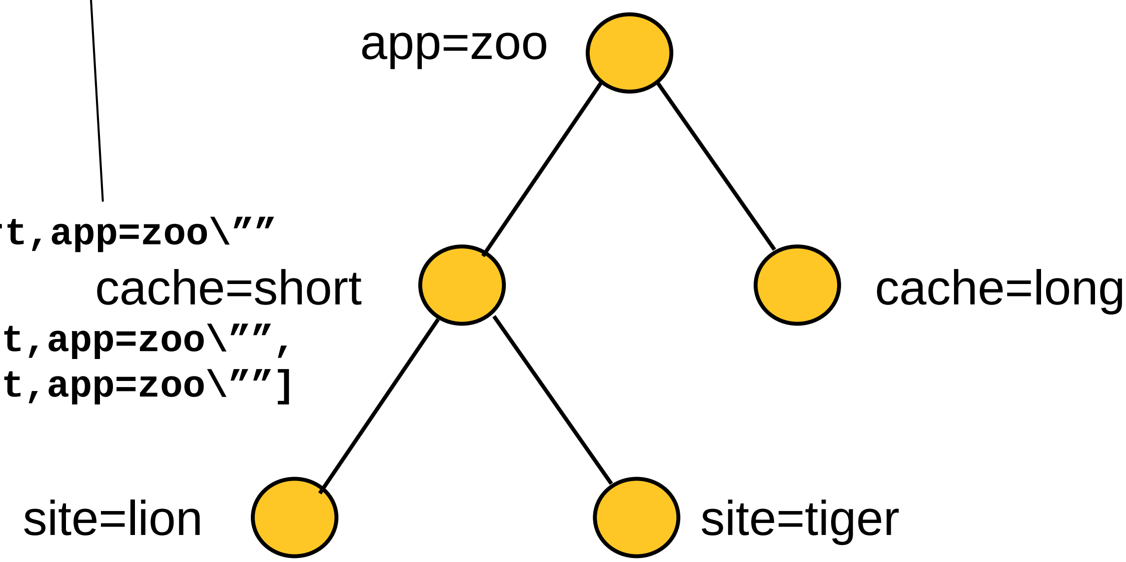
```
DefaultDomain:DN="site=lion,cache=short,app=zoo"
```

...key value pair
with the name of one object in the tree

Configuration Object Hierarchies

DN gives position in tree

```
ObjectName="DefaultDomain:DN=\"cache=short,app=zoo\""
children=
["DefaultDomain:DN=\"site=lion,cache=short,app=zoo\"",
"DefaultDomain:DN=\"site=tiger,cache=short,app=zoo\""]
```

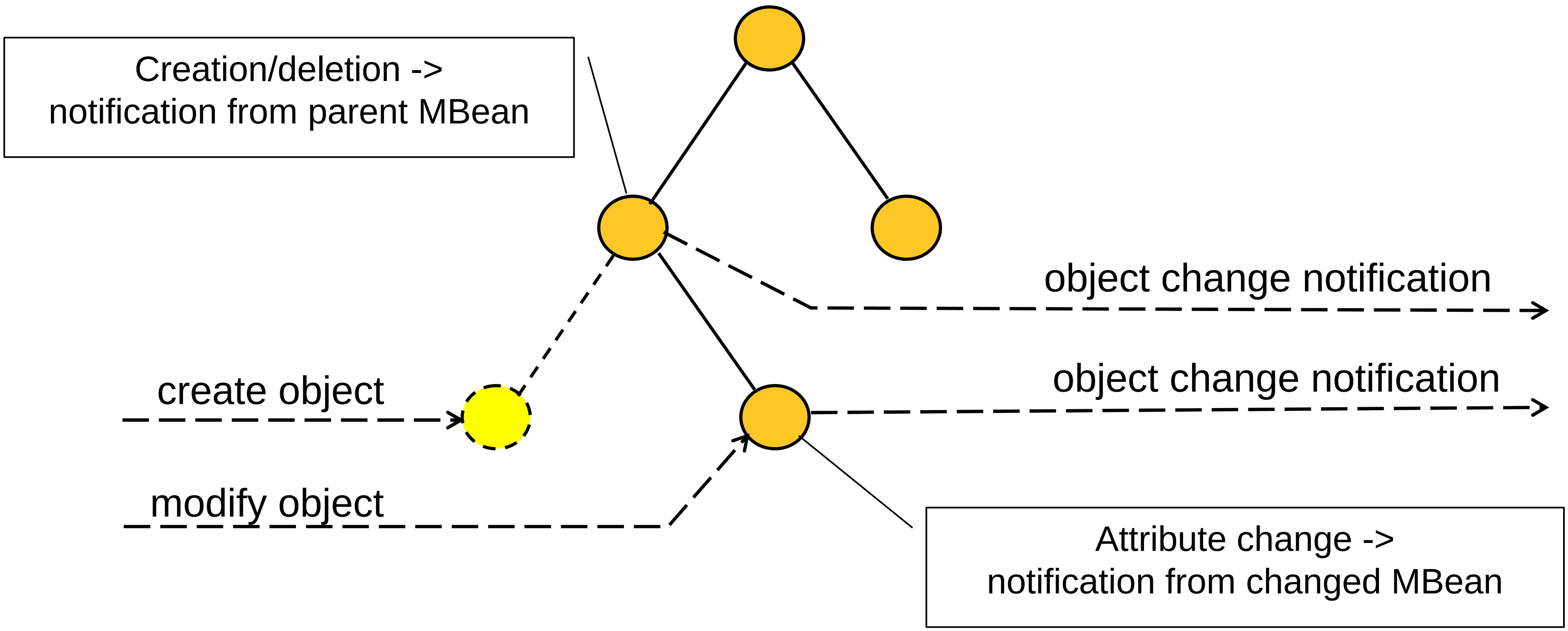


Attribute *children* makes tree navigatable

Configuration Change Notifications

- The JMX **`AttributeChangeNotification`**
  - reports one attribute only
- A JMX object change notification is defined by
  - using the generic **`Notification`**
  - requiring **`userData`** to be a **`CompositeData`** object
  - containing key value pairs with attribute names and values

Configuration Change Notifications



Configuration Validation

- Applications
  - optionally implements validator JavaBeans
  - associates bean classes with configuration object types
- Configuration provider
  - instantiates and populates beans
  - triggers beans to validate
  - rejects invalidated changes
- JSR 303 Bean Validation
  - has passed EDR
  - should be adopted when finalized

Agenda

- The Requirements
- A JMX Solution
- Code Samples
- Summary

Search the Object Tree

```
// Get an MBean server from the naming environment
@Resource(name="jmx_server")
MBeanServer mbs;

// Search all MBeans in a tree from a known root
// by using a wildcard in the distinguished name
Set<ObjectName> tree = mbs.queryNames(
    new ObjectName("DefaultDomain:DN=\"*app=zoo\""),
    null);
```

Read the Configuration

```
// Go through all "site" MBeans
for (ObjectName name: tree) {
    if (name.getKeyProperty("DN").
            startsWith("\"site")){
        String url = (String)
            mbs.getAttribute(name, "Url");
        // Set up connection to the URL
        ...
    }
}
```

Subscribe for Configuration Changes

```
// Go through all "site" MBeans
for (ObjectName name: tree) {
    if (name.getKeyProperty("DN").
            startsWith("\"site")){
    mbs.addNotificationListener(name, listener,
            null, null);
        String url = (String)
            mbs.getAttribute(name, "Url");
        // Set up connection to the URL
        ...
    }
}
```

Receive Configuration Changes

```
// Site change listener
public class SiteChangeListener
    implements NotificationListener {

    public void handleNotification(Notification ntf,
        Object handback) {
        ...
        CompositeData attributes = (CompositeData)
            ntf.getUserdata();
        if (attributes.containsKey(”Url”) {
            String url = (String) attributes.get(”Url”);
            // Change URL connection
            ...
        }
        ...
    }
}
```

Agenda

- The Requirements
- A JMX Solution
- Code Samples
- Summary

Summary

- JMX can provide configuration support
  - for clustered Java EE applications
  - in an easy way
- Problems solved:
  - atomic changes at runtime with notifications
  - object hierarchies
  - object validation
  - meta data available offline
  - efficient read access
- Solution: a configuration provider that
  - creates MBeans based on meta data
  - exposes configuration to applications and management systems
  - validates changes
  - handles persistence and distribution

THANK YOU

Peter Kristiansson
Senior Specialist
Ericsson

Jens Jensen
Expert
Ericsson

BOF-4945

ERICSSON
TAKING YOU FORWARD