
ORACLE®

ORACLE®

Java EE Configuration Service

Mike Keith
Oracle

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Some Common Requests and Use Cases

- Single application, multiple deployments
  - Don’t want to rip open application for each deployment
  - Multiple different resource definition packages
- Script-based configuration deployment
  - Don’t require manual deployment tool
- Deployment profiles
  - Development, Testing, Production, etc
- Environment properties 2.0
  - Available to one or to all apps
  - Simple to define and manage

Additional Requests

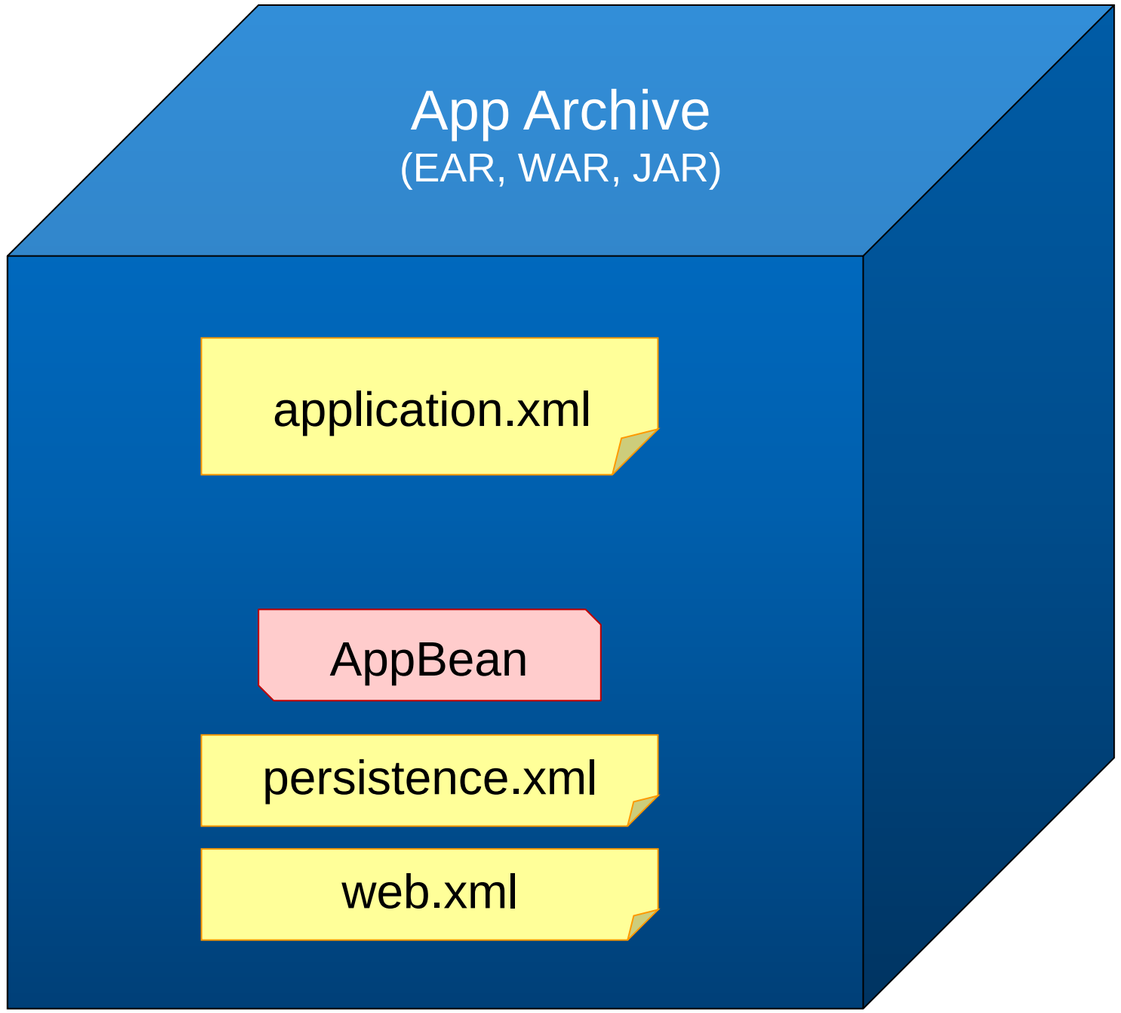
- Multiple modes of configuration access
  - Injection, integration with CDI
  - Java API for runtime access
- Versioned configuration
  - Dependency on a specific configuration version
- Dynamic SaaS tenant configuration
  - Admin wants to be able to add tenants (w/o restart)
  - SaaS application uses an API to look up a given tenant configuration
- Dynamic configuration change notification
  - Some apps want to be notified when a configuration change occurs

The Solution

- Add configuration support to Java EE
  - Configuration archives (e.g .car)
    - Created and deployed separately from application archives
    - Contain resource and property configuration definitions
  - Container integrates deployed configurations with existing services
  - Applications can declare dependencies on named configurations
  - Injection into Java EE managed components
  - Java API to access configured resources

Example – Application Development

```
<application … >
  <application-name>MyApp
  </application-name>
  ...
  <required-configs>
    <config
       name="myapp.config"/>
  </required-configs>
  ...
</application>
```



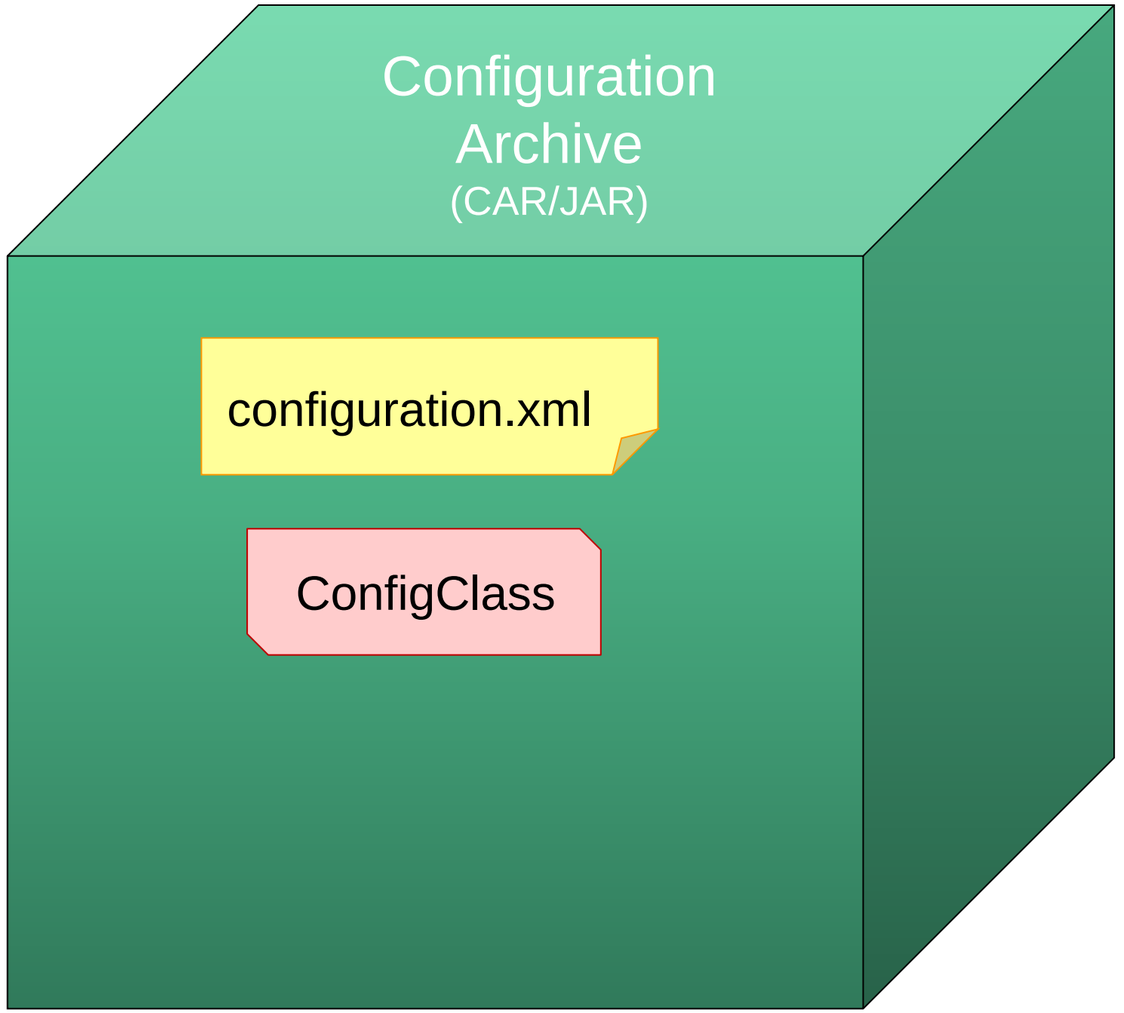

```
class AppBean {

  @Inject
  @Config("app.ds")
  DataSource someDS;
  ...
}
```

```
<persistence … >
  ...
  <jta-datasource>
     java:global/jdbc/myDS
  </jta-datasource>
  ...
</persistence>
```

```
<web-app … >
  ...
  <servlet-mapping>
     ...
    <url-pattern>${vendor}/*
    </url-pattern>
  </servlet-mapping>
  ...
</web-app>
```

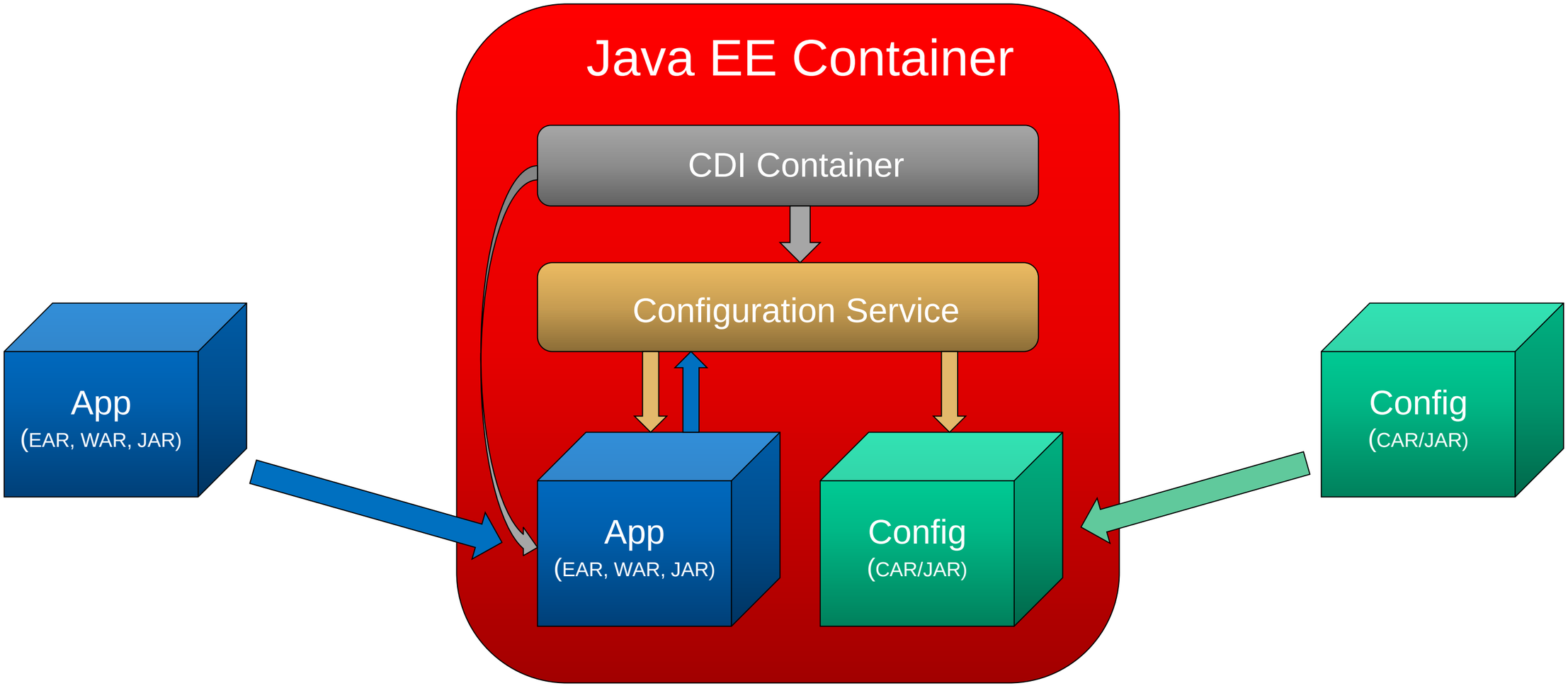
Example – Configuration Development

```
<configuration
   name="myapp.config“,
   scope=GLOBAL,
   profile=PRODUCTION,
   ... >
 <property name="app.ds">
  <data-source>
    <name>java:global/jdbc/myDS
    </name>
    <class-name>org…DataSource
    </class-name>
    <url>jdbc:…/myDB</url>
    ...
  </data-source>
 </property>
 <property name="vendor"
           value="acme"/>
</configuration>
```




```
@Configuration("myapp.config")
@ConfigProperty(name="app.ds",
 dataSource=
  @DataSourceDefinition(
  name="java:global/jdbc/MyDS",
  className="org…DataSource",
  url="jdbc:…/myDB",
  ...
  ))
@ConfigProperty(name="vendor",
                value="acme")
class ConfigClass { }
```

Example – Deployment and Runtime



Configuration API

- API used by applications to access configuration objects

```
interface ConfigurationService {

    Configuration getConfiguration(String configName);
    Collection<Configuration> getConfigurations(String nameFilter);

    Object getProperty(String propertyName);
    <T> T getProperty(String propertyName, Class<T> valueClass)
}

interface Configuration {

    String getName();

    String getProfile();

    boolean isGlobal();

    Object getProperty(String propertyName);
    <T> T getProperty(String propertyName, Class<T> valueClass)
}
```

Configuration API - Example

```
@ManagedBean
public class ApplicationBean {

  @Resource ConfigurationService configService;
  String vendor;

  @PostConstruct
  public void initialize() {
    vendor = configService.getProperty("vendor", String.class);
  }

  // ...

}
```

Feedback

- Do you experience any of the kinds of problems this service proposes to solve?
- If so, would this solution meet your need?
- Have you ever had a need for dynamic configuration (such as what OSGi might provide)?
- What kind of tool support would you expect to see for this?
- Do you have any different problems that you wish a service like this would solve?
- Other feedback?