

ORACLE



WebLogic to GlassFish



.consulting .solutions .partnership

Migration experiences
Markus Eisele
25/06/10

blog.eisele.net
@myfear



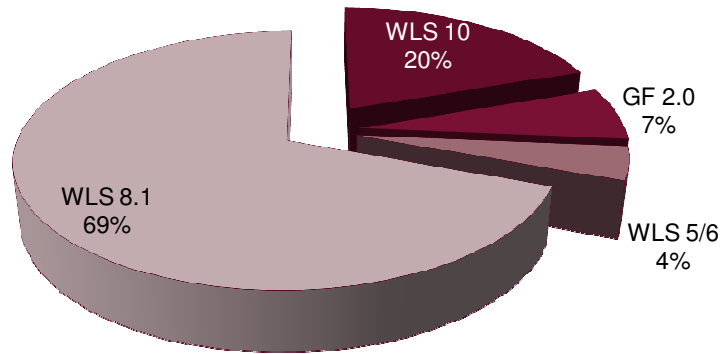
Preface.

.consulting .solutions .partnership



- This presentation sums up my experience in migrating roughly 20 WebLogic (8.1) applications to Java EE 5/GlassFish 2.
- I WILL talk about customer names but ask you NOT to do so if you refer to this talk anywhere!
- **The slides are NOT intended for publication or any use outside the Oracle ACE Directors circle!**

Number



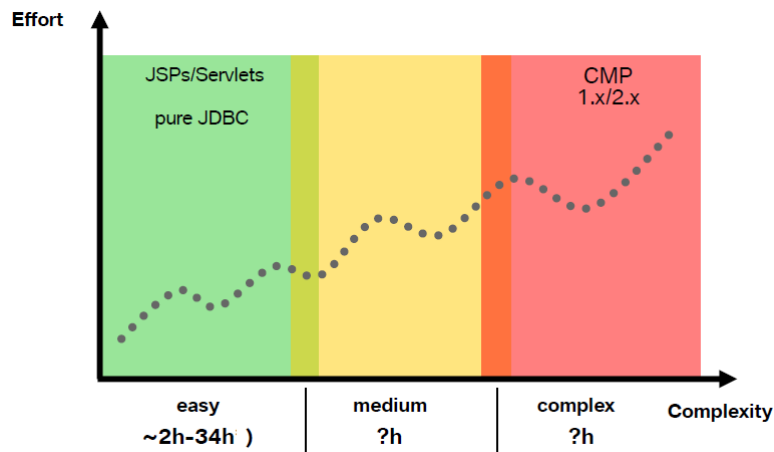
- General topics
 - Strategies
 - Complexity/Effort
 - Problems/Risks
- Technical topics
 - General Approach
 - Exemplary topics.

- **Strategy 1: Keep J2EE**
 - Replace Weblogic-specific features
 - comply with J2EE standard
 - rely on Java EE 5 backward compatibility
- **Strategy 2: Migrate to Java EE 5**
 - Replace Weblogic-specific features
 - change implementation to Java EE 5 (e.g. JPA, Annotations, ...)
 - comply with Java EE 5 specification
- **Strategy 3: Mix strategy 1 and 2 ;)**

- Complexity classes of applications
 - Dependency on application server
 - Architecture
 - Java/J2EE/Java EE
 - Persistency Layer
 - (Custom) Frameworks/Libraries
- Effort of migration depends on:
 - In general on complexity
 - E.g CMP/Entity Beans application
 - # and size of entities
 - # and complexity of queries
 - # of relationships/database structure

Effort.

consulting.solutions.partnership



© msg systems ag, Markus Eisele

7

Problems/Risks.

consulting.solutions.partnership



- WebLogic depended project
 - Use of proprietary classes/features
 - Weblogic-specific behavior
 - specific build/deployment
- Old frameworks
- Bad code quality
- Bad architecture
- Bad/no documentation

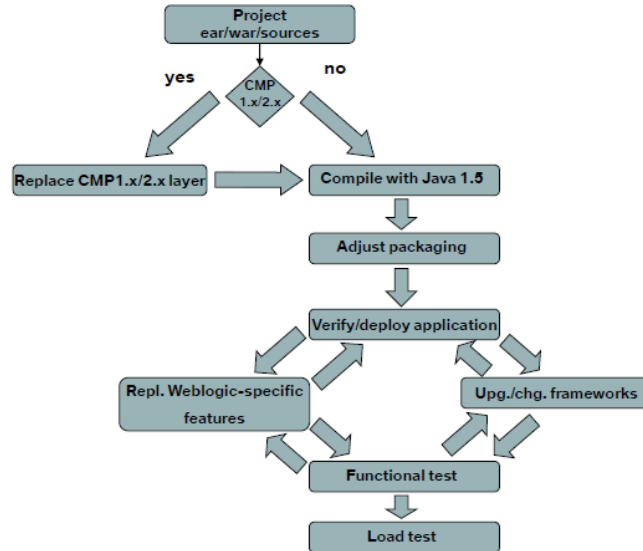
Many of the problems/risks also apply to any other migration project ;) !

© msg systems ag, Markus Eisele

8

General approach.

consulting.solutions.partnership



Basic migration requirements

consulting.solutions.partnership

- Basic requirements:
 - buildable project including all dependencies
 - Weblogic-based test environment
 - Non-critical test database,
 - Test cases, especially if CMP 1.x/2.x layer is replaced
 - Logins
 - Application URLs

Have a running application somewhere!

Replace CMP Layer.

- Transform Entity Beans in JPA entities
- Map JPA entities and their relations to database
- Reuse EJB QL as far as possible
- Reuse native queries (ejbHome methods)
- If used, find a migration strategy for value/transfer objects (optional)

Comply with \geq JDK 1.5

- Check code for incompatibilities/deprecations, e.g. enum used as variable name
- Align development environment with runtime environment
- A Java EE 5 project in development environment is recommended, if bigger code changes are necessary
- Resolve XML parser conflict (easy if JAXP is used)

Adjust packaging.

- Ear, jar and war files must be Java EE 5 (J2EE) compliant
- APP-INF/lib and APP-INF/classes are Weblogic-proprietary
- E.g. move libraries to \lib directory in ear
- If necessary move configuration files to a place where they are found (related to implementation and classloading)
- Remove Weblogic-specific jndi-properties

Additionally plan to spend some time with the build. Probably move to maven!

Replace Weblogic-specific features.

- Adjust URL patterns in web.xml
- Adjust welcome-file-list
- Remove Weblogic-specific JSP compilation features
- Remove Weblogic specific taglibs
- Add <ejb-ref> tags for EJB referencesExample

Upgrade/Change Frameworks.

- In general it's recommended to migrate to the current solutions
- Framework upgrades/changes are mostly independent from server migration
- With very little exceptions (struts \leq 1.0) you will be able to keep all existing frameworks (e.g. JCo, BIRT, Axis ...).

Verify/deploy application.

- Use the verifier to test application
 - NetBeans
 - Command line (glassfish-install/bin)
- Deploy application



.consulting .solutions .partnership