Java Enterprise Edition

Gabriele Tolomei

DAIS – Università Ca' Foscari Venezia

Administrivia

- Durata: **60 ore**
  - lezioni teoriche + esercitazioni pratiche
- Materiale Didattico
  - slides pubblicate sul sito web del corso
    http://gabrieletolomei.wordpress.com/teaching/yyyy-2014/java-ee/
  - risorse online
- Contatti:
  - gabriele.tolomei@unive.it

Requisiti

- Costrutti base di programmazione imperativa
  - if-else, for, while, etc.
- Paradigma di programmazione Object-Oriented (OO)
  - sintassi Java
- Fondamenti di computer networking
  - applicazioni client/server su Web
  - HTML/XML
  - protocollo HTTP (richiesta/risposta)

Programma del Corso

- 09/01 – Introduzione
- 10/01 – Java Servlets
- 16-17/01 – JavaServer Pages (JSP)
- 23-24/01 – Lab: Applicazione “AffableBean”
- 30-31/01 – Enterprise JavaBeans (EJB) + Lab

Modulo 1: Introduzione

- Tecnologia Java
  - 1 Linguaggio vs. 4 Piattaforme
- La piattaforma Java Enterprise Edition (Java EE)
  - Applicazioni Enterprise
- Java EE Application Servers
  - JBoss
- Ambiente di sviluppo
  - Eclipse

Modulo 1: Introduzione

- Tecnologia Java
  - 1 Linguaggio vs. 4 Piattaforme
- La piattaforma Java Enterprise Edition (Java EE)
  - Applicazioni Enterprise
- Java EE Application Servers
  - JBoss
- Ambiente di sviluppo
  - Eclipse

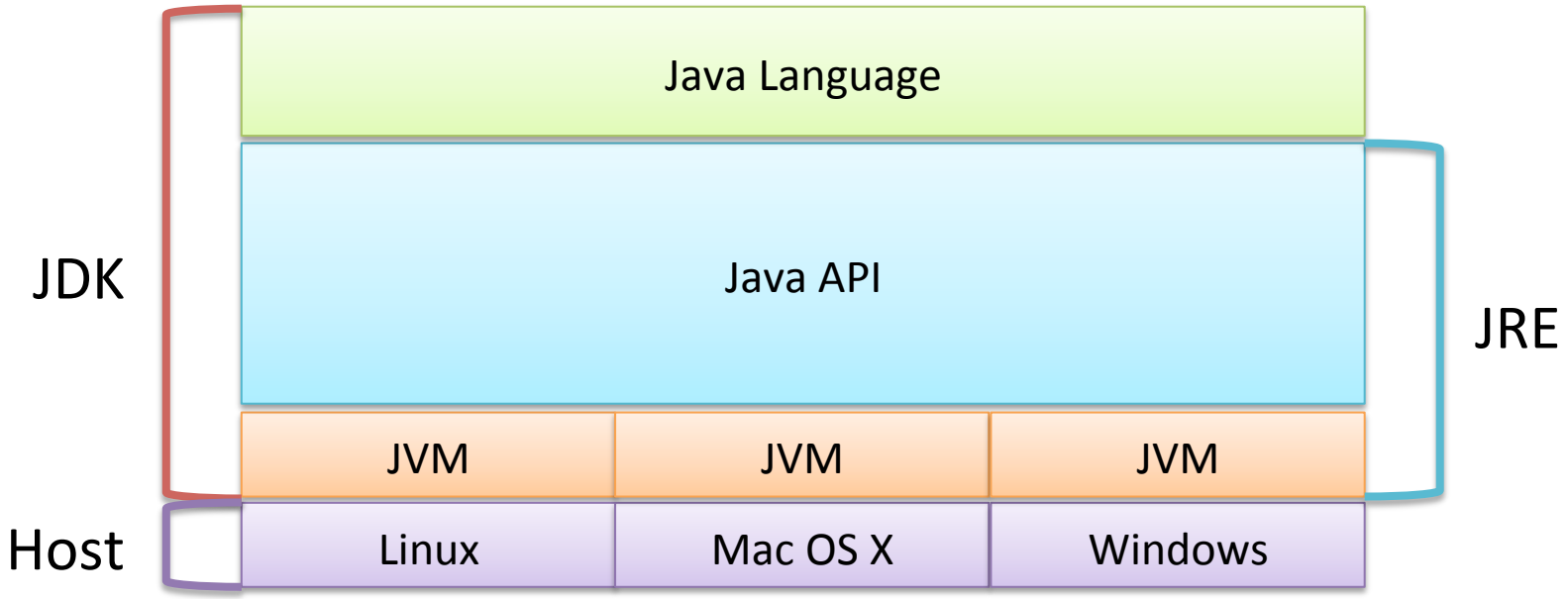
Tecnologia Java: Premessa

- Java è sia un **linguaggio di programmazione** che una **piattaforma**
  - il linguaggio di programmazione è un linguaggio "ad alto livello" che supporta il paradigma OO
  - la piattaforma specifica l'ambiente in cui le applicazioni (programmi) scritte in Java vengono eseguite (**Java Runtime Environment** o **JRE**)
- 1 Linguaggio di Programmazione vs. 4 Piattaforme
  - Java Standard Edition (Java SE)
  - Java Enterprise Edition (Java EE)
  - Java Micro Edition (Java ME)
  - JavaFX

Le Piattaforme Java

- Tutte le piattaforme Java consistono di
  - **Java Virtual Machine** (**JVM**) + **Application Programming Interface** (**API**)
- La **JVM** è un particolare programma (**interprete**) eseguito su uno specifico sistema ospite (**host**) che consente l'esecuzione di programmi Java
  - JVM interpreta il codice intermedio (**bytecode** contenuto in file .class) risultato della compilazione del codice sorgente (.java)
- Esistono varie implementazioni JVM, una per ciascun sistema host supportato
  - Linux x86/x64
  - Mac OS X x64
  - Win x86/x64
  - …
- La **API** è una collezione di componenti software "standard" messi a disposizione degli sviluppatori Java per creare nuovi componenti e/o applicazioni

Java: Linguaggio + Piattaforma



Java: Vantaggi

- **Portabilità** (“Write-Once-Run-Anywhere”)
  - Applicazione sviluppata in accordo alle specifiche di una piattaforma Java e compilata su un determinato host
  - Applicazione eseguita su qualsiasi altro host purché questo fornisca un’implementazione della stessa piattaforma (JVM + API)
  - Ad es. applicazione Java SE sviluppata/compilata su Windows ed eseguita su Linux
- **Facilità** di sviluppo software
- **Sicurezza**
- …

Java SE

- È la piattaforma di riferimento quando si parla di Java
- Java SE API fornisce le funzionalità “core” del linguaggio
  - tipi nativi (ad es., int, boolean, char, etc.)
  - classi e oggetti base (ad es., Class, Object, String, etc.)
  - classi e oggetti per gestire
    - I/O
    - Security
    - Database
    - Graphical User Interface (GUI)
    - XML
    - …

Modulo 1: Introduzione

- Tecnologia Java
  - 1 Linguaggio vs. 4 Piattaforme
- La piattaforma Java Enterprise Edition (Java EE)
  - Applicazioni Enterprise
- Java EE Application Servers
  - Jboss
- Ambiente di sviluppo
  - Eclipse

Java EE

- Realizza una piattaforma “standard” per lo sviluppo, l’esecuzione e la gestione di **applicazioni enterprise**:
  - **Multi-tier** → strutturate a “livelli”
  - **Web-enabled** → accessibili via Web
  - **Server-centric** → eseguite in uno specifico ambiente server
  - **Component-based** → costituite da componenti sw in esecuzione su una o più istanze server distribuite
- Si basa sulla piattaforma Java SE a cui aggiunge specifiche e strumenti (API) ad hoc
- Condivide i vantaggi delle applicazioni Java SE:
  - 1 specifica standard vs. molte implementazioni
  - implementazioni disponibili per la maggior parte di sistemi host
  - portabilità, facilità di sviluppo, riuso, sicurezza, etc.

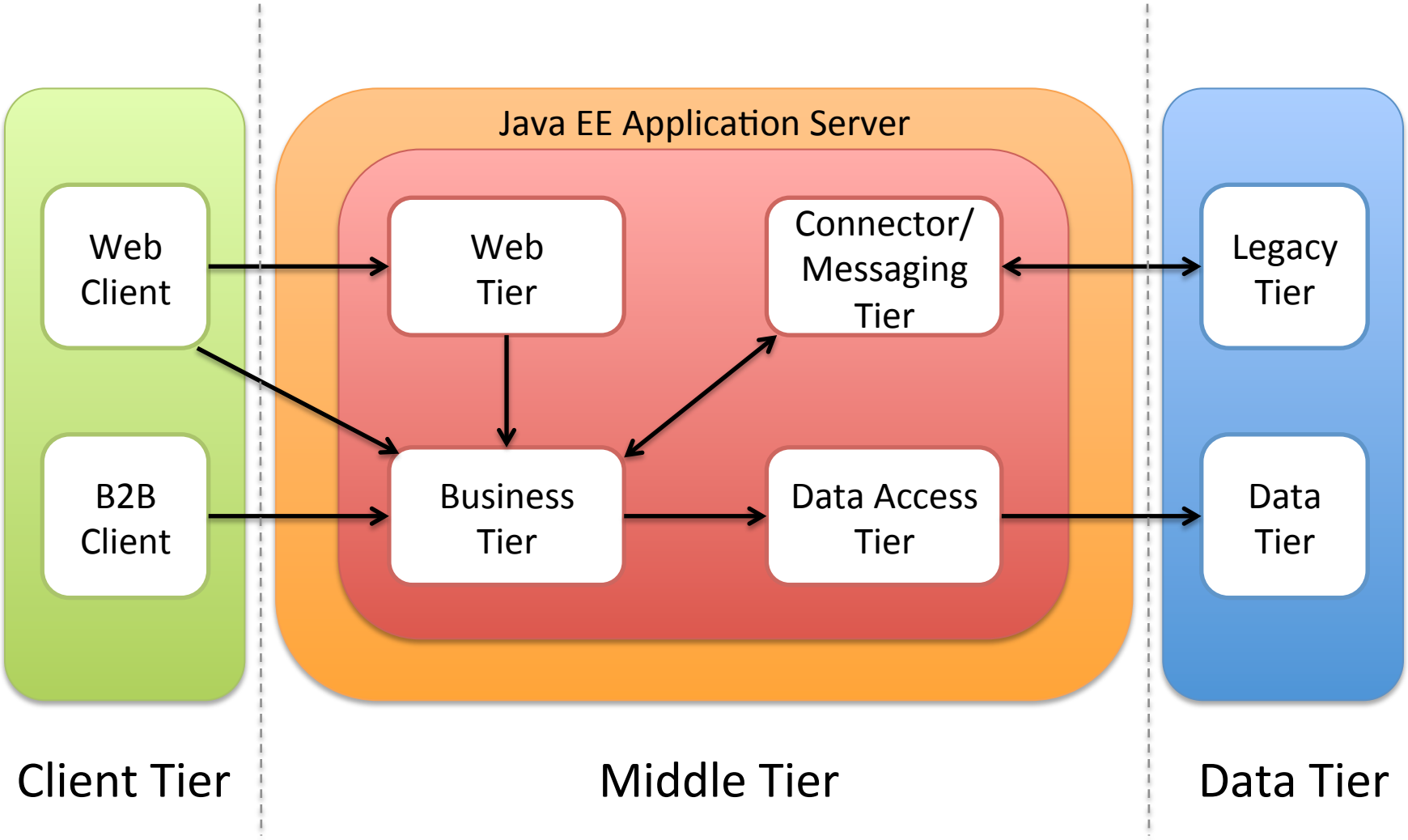
Applicazioni Enterprise

- Applicazioni progettate e sviluppate per risolvere problemi tipici delle “grandi aziende”
- Utili anche per piccole imprese/organizzazioni che sfruttano l’altissimo grado di connettività ormai raggiunto
- Esempi:
  - E-commerce
  - Online Banking
  - …

Architettura Multi-tier

- **Modello** architetturale "astratto" per applicazioni enterprise
  - indipendente dalle scelte tecnologiche (linguaggio, piattaforma, etc.)
- Le funzionalità dell'applicazione sono suddivise in 3 "livelli" isolati (**Tiers**):
  - **Client Tier** → esegue richieste al Middle-tier
  - **Middle Tier** → gestisce le richieste provenienti dai clients e processa i dati dell'applicazione
  - **Data Tier** → mantiene i dati in strutture di memorizzazione permanenti
- Java EE è una particolare realizzazione del modello che si concentra sul Middle Tier → Java EE Application Server

Java EE: Architettura Multi-tier



Java EE: Client Tier

- Il **Client Tier** comprende gli applicativi client che "utilizzano" l'applicazione enterprise comunicando con il Java EE Application Server
- I clients sono di solito in esecuzione su hosts diversi da quello che ospita il server
- 2 tipi di applicativi client:
  - **Web Client** → un browser web che esegue richieste via HTTP al Web Tier
  - **B2B Client** → una o più applicazioni che eseguono richieste al Business Tier tramite SOAP/Web Services o Java RMI

Java EE: Web Tier

- Il **Web Tier** consiste di componenti che gestiscono le interazioni tra i Web client e il **Business Tier**
- Funzioni principali:
  - generazione dinamica ("on-the-fly") dei contenuti per i diversi client
  - raccolta dati di input che gli utenti inviano tramite interfaccia Web client
  - generazione output sulla base delle componenti del Business Tier
  - controllo del flusso di navigazione sul client
  - mantenimento dello stato per una sessione utente
  - logica applicativa di base e memorizzazione temporanea di informazione all'interno di componenti Java (JavaBeans)

Java EE: Web Tier

Tecnologia	Scopo
Servlets	Classi Java che processano le richieste HTTP e generano dinamicamente le risposte (HTML)
JavaServer Faces	Framework per il design dell'interfaccia utente di applicazioni Web
JavaServer Faces Facelets	Particolari applicazioni JavaServer Faces che usano pagine XHTML anziché JSP
Expression Language	Insieme di tags standard usati in JSP e Facelets per riferirsi a componenti Java EE
JavaServer Pages (JSP)	Documenti testuali compilati e trasformati in Servlets per aggiungere contenuto dinamico a pagine HTML
JavaServer Pages Standard Tag Library	Tag library che raccoglie funzionalità comuni a pagine JSP
JavaBeans Components	Oggetti Java per la memorizzazione temporanea dei contenuti di un'applicazione

Java EE: Business Tier

- Il **Business Tier** consiste di componenti che forniscono la cosiddetta *business logic* dell'applicazione
- Per "business logic" si intende l'insieme del software che si occupa delle funzionalità di un determinato contesto di business
- Costituisce il "core" dell'intera applicazione in quanto vero e proprio responsabile della fase di processing

Java EE: Business Tier

Tecnologia	Scopo
Enterprise JavaBeans (EJB)	Componenti gestite dall'Application Server che incapsulano le funzionalità principali dell'applicazione
JAX-RS RESTful Web Services	API per la creazione di Web Services REST (via HTTP GET e POST)
JAX-WS Web Service Endpoints	API per la creazione ed il consumo di Web Services XML/SOAP
Java Persistence API Entities	API per il mapping tra i dati contenuti nei sistemi di memorizzazione persistente e corrispondenti oggetti Java
Java EE Managed Beans	Essenzialmente EJB che non richiedono requisiti di sicurezza/transazionalità

Java EE: Data Tier

- Il **Data Tier** si riferisce alle varie "sorgenti dati" cui può attingere l'applicazione e comprende:
  - Relational Database Management Systems (MySQL, Oracle, etc.)
  - Enterprise Resource Planning Systems (SAP)
  - Mainframes (IBM AS/400)
- Le sorgenti dati
  - sono localizzate su hosts diversi da quello su cui è in esecuzione il Java EE Application Server
  - vengono accedute dalle componenti del Business Tier

Java EE: Data Tier

Tecnologia	Scopo
Java Database Connectivity API (JDBC)	API a basso livello per l'accesso ed il recupero dei dati memorizzati su supporti permanenti. Tipicamente usata per eseguire query SQL ad un particolare RDBMS
Java Persistence API	API per la creazione di Web Services REST (via HTTP GET e POST)
Java EE Connector Architecture	API per la creazione ed il consumo di Web Services XML/SOAP
Java Transaction API (JTA)	API per la definizione e la gestione delle transazioni tra sorgenti dati multiple e distribuite

Modulo 1: Introduzione

- Tecnologia Java
  - 1 Linguaggio vs. 4 Piattaforme
- La piattaforma Java Enterprise Edition (Java EE)
  - Applicazioni Enterprise
- Java EE Application Servers
  - JBoss
- Ambiente di sviluppo
  - Eclipse

Java EE Application Servers

- Server che **implementa** la piattaforma **Java EE**
- Ospita i componenti **Middle Tier** di un'applicazione enterprise multi-tiered
- Fornisce i **servizi standard** specificati da Java EE a questi componenti sottoforma di **container**:
  - gestione della concorrenza, scalabilità
  - sicurezza
  - persistenza, transazioni
  - gestione del ciclo di vita dei componenti sw
- Java EE servers "famosi": GlassFish (Oracle), JBoss AS (Red Hat), WebLogic (Oracle-BEA), WebSphere (IBM), etc.
  - http://en.wikipedia.org/wiki/Comparison_of_application_servers#Java

Java EE Containers

- Interfaccia tra un componente dell'applicazione e le funzionalità di "basso livello" fornite dalla piattaforma per supportare quel componente
- Le funzionalità di un container sono specificate dalla piattaforma
- Un tipo di container per ciascun tipo di componente
- Java EE Server fornisce ai vari containers un ambiente omogeneo in cui è garantito il funzionamento di ciascun componente dell'applicazione

Web Container

- Interfaccia tra le componenti web ed il server web
- Un componente web può essere una **Servlet**, una pagina **JSF** o **JSP**
- Gestisce il ciclo di vita del componente
- Smista le richieste ai vari componenti dell'applicazione
- Fornisce interfacce verso "dati contestuali" (ad es. informazioni sulla richiesta corrente)

Application Client Container

- Interfaccia (**gateway**) tra le **applicazioni** client Java EE ed il **server** Java EE
- I clients sono particolari applicazioni Java SE che usano i componenti server Java EE
- In esecuzione su macchine client (generalmente diverse dal server Java EE)

EJB Container

- Interfaccia tra Enterprise JavaBeans che implementano la business logic dell'applicazione e il server Java EE
- In esecuzione sulla macchina che ospita il server Java EE
- Gestisce l'esecuzione dei componenti EJB dell'applicazione

Java EE Server: JBoss

- Implementazione **open source** delle specifiche Java EE
  - **JBoss Enterprise Middleware Suite** (**JEMS**)
- **JEMS** contiene:
  - Java EE Application Server (JBoss AS, Tomcat)
  - O/R Mapping e Persistence (Hibernate)
  - Portal Platform (JBoss Portal)
  - Business Process Management and Rules (JBoss jBPM, JBoss Rules)
  - Object/Data Cache (JBoss Cache)
  - Distributed Transaction Management (JBoss Transactions)
  - Development Tools (JBoss Tools plugin for Eclipse)

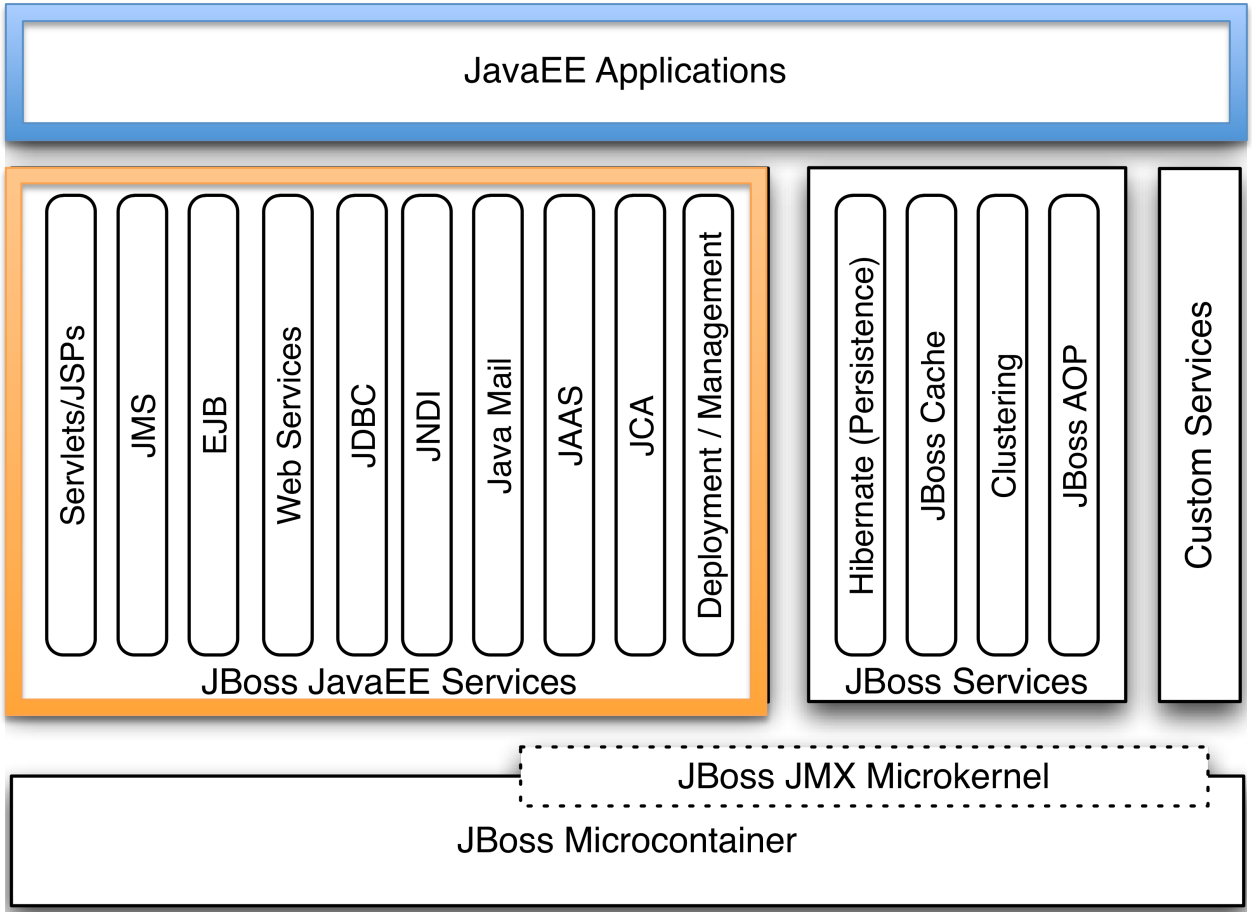
JBoss AS: Un po' di storia…

- Progetto open source di un EJB Container (1999)
- Supporto di J2EE server (ver. 2.x)
- Service-oriented Architecture (ver. 3.x)
- Supporto di Java EE 1.4 + EJB 3 (ver. 4.x)
- Supporto di Java EE 5 (ver. 5.x)
- Supporto di Java EE 6 (ver. 6.x)
- Versione attuale: JBoss 7.1.1 → WildFly 8

JBoss AS 5.1

- Java EE Application Server che useremo durante il corso
- **JBoss Web** → Apache Tomcat 6.0
- **JBoss WS 3.0** → Supporto Web Services per JAX-WS/JAX-RPC
- 2 nuove configurazioni:
  - **standard**: fully Java EE
  - **web**: Java Servlet/JSP container + JTA/JCA + JPA (il server può essere acceduto solo via HTTP)
- Riferimento documentazione:
  - http://www.jboss.org/jbossas/docs/5-x.html

JBoss AS 5.1: Architettura



JBoss AS 5.x: Requisiti di Sistema

	JBoss 5.x
Java SE (JDK)	1.5+
OS/Platform	Any Java-compliant
Main Memory (RAM)	512 MB
CPU	400 MHz
Disk	100 MB
DB (optional)	Any JDBC-compliant

NOTA: JBoss AS non richiede l'intero JDK, per cui il solo JRE è sufficiente per un corretto funzionamento delle applicazioni in ambiente di produzione.
Tuttavia, JDK è utile e raccomandato perché fornisce tools extra di supporto

JBoss AS 5.1: Directories

- La directory “root” di installazione è riferita come **$JBOSS_HOME** (variabile di ambiente) o **jboss.home.dir** (Java system property) e contiene:

```
JBossORG-EULA.txt       copyright.txt           lib/
bin/                    docs/                   readme.html
client/                 jar-versions.xml        server/
common/                 lgpl.html
```

- Importante conoscere il layout delle directories per:
  - localizzare le librerie
  - aggiornare la configurazione del server
  - eseguire il deployment di applicazioni e servizi

JBoss AS 5.1: Directory bin

```
README-service.txt      run.conf              twiddle.sh
classpath.sh            run.conf.bat          wsconsume.bat
jboss_init_hpux.sh      run.jar               wsconsume.sh
jboss_init_redhat.sh    run.sh                wsprovide.bat
jboss_init_suse.sh      service.bat           wsprovide.sh
jbosssvc.exe            shutdown.bat          wsrunclient.bat
password_tool.sh        shutdown.jar          wsrunclient.sh
probe.bat               shutdown.sh           wstools.bat
probe.sh                twiddle.bat           wstools.sh
run.bat                 twiddle.jar
```

JBoss AS `${jboss.home.dir}/bin` directory contains startup/shutdown scripts, bootstrap libraries, Web Services and server management utilities:

- `classpath.sh`: A tool to determine JBoss classpaths (both client and server)
- `jboss_init_redhat.sh` and `jboss_init_suse.sh`: JBoss system control scripts for RedHat and SuSE systems
- `probe.sh` and `probe.bat`: used for discovering JBoss AS clusters.
- `run.sh` and `run.bat`: Scripts for starting JBoss AS
- `run.jar`: Bootstrap code for starting JBoss AS
- `service.bat`: Script to manage JBoss as a Windows service.
- `shutdown.sh` and `shutdown.bat`: Scripts for shutting down JBoss AS (including remote instances)
- `shutdown.jar`: Bootstrap code for shutting down JBoss AS
- `twiddle.sh` and `twiddle.bat`: Scripts for running JBoss AS command-line management client (based on JMX)
- `twiddle.jar`: Bootstrap code for the JMX management (instrumentation) client
- `wsconsume`, `wsprovide`, `wsrunclient` and `wstools` are utilities for Web Services. We will see that later on Web Services chapter.

JBoss AS 5.1: Directory client

- Contains the Java libraries (JARs) required for clients that run outside the JBoss AS containers, such as:
  - WebService clients
  - EJB clients
  - JMX clients
- Used by external applications that need to access JNDI resources
- On Unix, to get the client CLASSPATH, run: `${jboss.home.dir}/bin/classpath.sh -c`
- As of JBoss 5, the file `client/jbossall-client.jar` contains references to other JARs via `Class-Path` setting in its `META-INF/MANIFEST.MF` file. This makes it possible for external JBoss clients to just reference this one JAR file as opposed to many of them.

JBoss AS 5.1: Directory common

- Contains the `lib` directory (also known as `jboss.common.lib.url`).
  - `lib` folder contains the common libraries shared by all server configurations (more on this later)
  - This directory is new to JBoss 5. In earlier versions of JBoss a number of common libraries were simply duplicated for each configuration set.

JBoss AS 5.1: Directory docs

```
dtd/            examples/       licenses/       schema/         tests/
```

- Examples: sample configuration: JMX, JCA, JMS, NetBoot, etc.
  - Contains excellent examples for many different configurations. For example, the file `docs/examples/jca/mysql-ds.xml` can serve as a starting point in defining a MySQL-based DataSource (shared database connection-pool) in JBoss AS.
- DTDs and Schemas for J2EE and JBoss XML files
  - Contain all J2EE-referenced XML DTD and XSD files, making it simple to validate XML files and lookup the relevant "grammar" when making configuration changes. DTDs were employed by J2EE 1.3 and JBoss 4 whereas Schemas are used since J2EE 1.4 and JBoss 5.
- Licenses for all JBoss components
- Unit test results

JBoss AS 5.1: Directory lib

- Contains JBoss bootstrap libraries (core libraries)
- Do not place your own files here or remove any of the existing files
- As an example, you'll find here the JBoss Microcontainer and the old JMX kernel.

JBoss AS 5.1: Directory server

```
all/            minimal/        node2/          web/
default/        node1/          standard/
```

- Known in JBoss AS as `jboss.server.base.dir`
- Root of server configuration sets
- JBoss comes with `minimal`, `default` and `all`
- Version 5.x comes with 2 new configurations: `standard` and `web`
- Defaults to configuration set in `server/default`
- Configuration sets contain the actual JBoss services

To change the configuration set that JBoss AS runs with, execute: `bin/run.sh -c <configuration-set>`

For example:

```
bin/run.sh -c minimal
```

```
bin/run.sh -c all
```

Configuration sets:

minimal/

- Includes support for JNDI and logging. It does not contain any other J2EE services like Servlet/JSP container, EJB container, or JMS.
- Can serve as a starting point when creating your own configuration sets

default/

- As the name implies, this is the default Java EE 5 configuration. Contains the most used services except JAXR, IIOP and clustering services.

all/

- This configuration extends the default configuration set and also include JAXR, IIOP and clustering services

standard/

- Certified Java EE 5 configuration compliant.

web/

- Lightweight web container profile (Java EE 6 web profile). It provides support for JTA/JCA and JPA except for the servlet/JSP container.

JBoss AS 5.1: Directory default/conf

```
bindingservice.beans/          jbossts-properties.xml
bootstrap/                     jndi.properties
bootstrap.xml                  login-config.xml
java.policy                    props/
jax-ws-catalog.xml             standardjboss.xml
jboss-log4j.xml                standardjbosscmp-jdbc.xml
jboss-service.xml              xmdesc/
```

- Known in JBoss as `jboss.server.config.url`
- Contains a bootstrap descriptor (`jboss-service.xml`) that defines which services are loaded for the lifetime of the instance

The files in directory `${jboss.server.config.url}` :

- `bootstrap/*`: Bootstrap descriptors for core microcontainer services defined in `bootstrap.xml`
- `bootstrap.xml`: Defines the core microcontainer beans to load during bootstap
- `jboss-service.xml`: Defines the core JMX services configurations
- `jndi.properties`: Specifies a set of properties that are passed to JNDI when `new InitialDirContext()` is called within JBoss
- `jboss-log4j.xml`: Configuration file for the logging service (Log4J) defining log filters, priorities, and destinations
- `login-config.xml`: Defines security realms used for authentication and authorization (JAAS)
- `props/*.properties`: Java property files (usually used for JAAS realms)
- `java.policy`: Placeholder for security permissions (Java Security Manager). Grant-All by default
- `standardjboss.xml`: Configuration file for the standard EJB container
- `standardjbosscmp-jdbc.xml`: Configuration file for the standard JBossCMP engine
- `xmdesc/*-mbean.xml`: XMBean descriptors for services configured in the `jboss-service.xml` file
- `props/*`: property files defining users and roles for the `jmx-console`

Important

Any changes to files in this directory require a full server restart in order to take effect.

JBoss AS 5.1: Directory default/data

```
hypersonic/             wsdl/
tx-object-store/        xmbean-attrs/
```

- Known in JBoss as `jboss.server.data.dir`
- Location where some services store private content on the file system
  - Hypersonic DB - built-in (by default use as the temporary message store by JMS)
  - XMBeans attribute persistence (not enabled by default)
  - Transaction objects (temporary storage of objects during the two-phase commit process)
- This directory is not directory exposed to the end users (e.g. though the web interface)

Note

Unless you use Hypersonic DB, the contents of this directory (including the directory itself) can be cleared (deleted) between JBoss restarts.

JBoss AS 5.1: Directory default/deploy

```
CurrencyConverterApp.ear
ROOT.war/
admin-console.war/
cache-invalidation-service.xml
ejb2-container-jboss-beans.xml
ejb2-timer-service.xml
ejb3-connectors-jboss-beans.xml
ejb3-container-jboss-beans.xml
ejb3-interceptors-aop.xml
ejb3-timerservice-jboss-beans.xml
fortune.war/
hdscanner-jboss-beans.xml
hsqldb-ds.xml
http-invoker.sar/
jboss-local-jdbc.rar
jboss-xa-jdbc.rar
jbossweb.sar/
jbossws.sar/
jca-jboss-beans.xml
jms-ra.rar
jmx-console.war/
jmx-invoker-service.xml
jmx-remoting.sar/
jsr88-service.xml
legacy-invokers-service.xml
mail-ra.rar
mail-service.xml
management/
messaging/
monitoring-service.xml
my-ws.war
printservice.sar/
profileservice-jboss-beans.xml
profileservice-secured.jar/
properties-service.xml
quartz-ra.rar
remoting-jboss-beans.xml
schedule-manager-service.xml
scheduler-service.xml
security/
sqlexception-service.xml
transaction-jboss-beans.xml
transaction-service.xml
uuid-key-generator.sar/
vfs-jboss-beans.xml
xnio-provider.jar/
```

- Dynamic deployment content directory
- This is where applications and services are deployed
- Default location used by hot deployment service
- Contains code and configuration files for all services

Some files in the `deploy` directory include:

- `ROOT.war`: is the / root web application
- `cache-invalidation-service.xml` - Custom invalidation of EJB caches via JMS
- `ejb2-container-jboss-beans.xml` - UserTransaction integration bean for EJB2 container
- `ejb2-timer-service.xml` - EJB timer service bean
- `ejb3-connectors-jboss-beans.xml` - EJB3 remoting transport bean
- `ejb3-container-jboss-beans.xml` - UserTransaction integration bean for EJB3 container
- `ejb3-interceptors-aop.xml` - AOP for EJB3
- `ejb3-timer-service.xml` - alternate quartz based timer service
- `hdscanner-jboss-beans.xml` - hot deployment scanner bean
- `hsqldb-ds.xml` - Hypersonic embedded database and related connection factories
- `http-invoker.sar` - Detached invoker that supports RMI/HTTP
- `jboss-local-jdbc.rar` - JCA to DataSource adaptor for JDBC drivers
- `jboss-xa-jdbc.rar` - JCA to XADataSource adaptor for JDBC drivers.
- `jbossweb.sar` - Tomcat Servlet/JSP Engine
- `jbossws.sar` - The JBoss service that supports web services
- `jca-jboss-beans.xml` - Connection management facilities for integrating JCA adaptors into JBoss AS
- `jms-ra.rar` - JMS resssource adapter
- `messaging/connection-factories-service.xml` - DLQ, ExpiryQueue JMS connection factory
- `messaging/destinations-service.xml` - Message persistence store service
- `messaging/jms-ds.xml`
- `messaging/legacy-service.xml`
- `messaging/messaging-jboss-beans.xml` - JMS security and management beans
- `messaging/messaging-service.xml` - Core messaging service
- `messaging/remoting-bisocket-service.xml` - JMS remoting service layer

JBoss AS 5.1: Directory default/deployers

```
alias-deployers-jboss-beans.xml
bsh.deployer/
clustering-deployer-jboss-beans.xml
dependency-deployers-jboss-beans.xml
directory-deployer-jboss-beans.xml
ear-deployer-jboss-beans.xml
ejb-deployer-jboss-beans.xml
ejb3.deployer/
hibernate-deployer-jboss-beans.xml
jboss-aop-jboss5.deployer/
jboss-ejb3-endpoint-deployer.jar
jboss-jca.deployer/
jboss-threads.deployer/
jbossweb.deployer/
jbossws.deployer/
jsr77-deployers-jboss-beans.xml
logbridge-jboss-beans.xml
messaging-definitions-jboss-beans.xml
metadata-deployer-jboss-beans.xml
seam.deployer/
security-deployer-jboss-beans.xml
webbeans.deployer/
xnio.deployer/
```

- Contains all the JBoss AS services that are used to recognize and deploy different application and archive types.

Some files in the `deployers` directory:

- `hibernate-deployer-jboss-beans.xml` - Deployer for Hibernate archives (HAR)
- `ejb-deployer-jboss-beans.xml` - Service responsible for deploying EJB JAR files
- `ear-deployer-jboss-beans.xml` - Service responsible for deploying EAR files
- `jbossweb.deployer` - Service responsible for deploying WAR files
- `jboss-aop-jboss5.deployer` - Deployer that sets up Aspect Manager Service and deploys AOP applications
- etc...

Note

This directory is new as of JBoss AS 5

JBoss AS 5.1: Directory default/lib

- Directory referred to by the bootstrap code when loading the configuration set
- Known within JBoss as `jboss.server.lib.url`
- This directory is for Java code (JARs) to be used both by the deployed applications and JBoss AS services
- If you have Java libraries that you need to be made available to all your applications/services, these can be placed in the `${jboss.server.lib.url}` directory.
- Similarly, you would also use this directory for Java libraries that need to be used by both your applications/services, and JBoss AS services.
    - A typical example of this is a JDBC driver that is needed by JBoss AS to manage a pool of database connections, as well as your code, which implicitly uses it to interact with the database server.

Note

As of JBoss 5, some JARs that used to reside in this directory have been moved to `common/lib` in order to share them with other configuration sets.

JBoss AS 5.1: Directory default/log

- Known within JBoss as `jboss.server.log.dir`
- Default destination directory for JBoss AS log files (3 log files)
- `boot.log` - logs boot process until logging service starts
- `server.log` - takes over once the logging service is initialized from `${jboss.server.config.url}/jboss-log4j.xml`
- `audit.log` - audit security
- Default startup log priority: `DEBUG`
- `STDOUT` and `STDERR` are logged to console

By default:

- Log file `server.log` is rolled over daily (with the ".yyyy-MM-dd" extension)
- Existing logs are overwritten on [re]start
- Old log files are not automatically cleaned by the server during runtime

Since the logging system is managed by Log4J it can be easily configured to:

- Roll over logs hourly
- Roll over logs by size (e.g. 500KB)
- Automatically remove old logs
- Log to SMTP, SNMP, Syslog, JMS, etc.

This directory can be cleared (deleted) between JBoss restarts.

JBoss AS 5.1: Directory default/tmp

- Known in JBoss as `jboss.server.temp.dir`
- Used by JBoss AS to store temporary files such as unpacked service and application deployments
- Deployments are automatically removed on server shutdown

This directory can be cleared (deleted) between JBoss restarts.

Note

Unpacked deployments (e.g. expanded WAR files) are not copied over. Packed deployments (WAR, EAR, RAR) are uncompressed, whereas JARs and XML-described services are copied over.

JBoss AS 5.1: Directory default/work

- Directory where compiled JSP `.java` and `.class` files reside
- Also contains cached TLDs
- Very useful for debugging problems in JSPs

Java ServerPages (`.jsp` files) are automatically compiled into Java Servlets (`.java` file) and then into Java byte-code (`.class` files) by Tomcat (the embedded servlet engine running within JBoss AS).

Many JSP errors are easier to fix when developers are able to look at the compiled `.java` files and match the line numbers to error/exception messages.

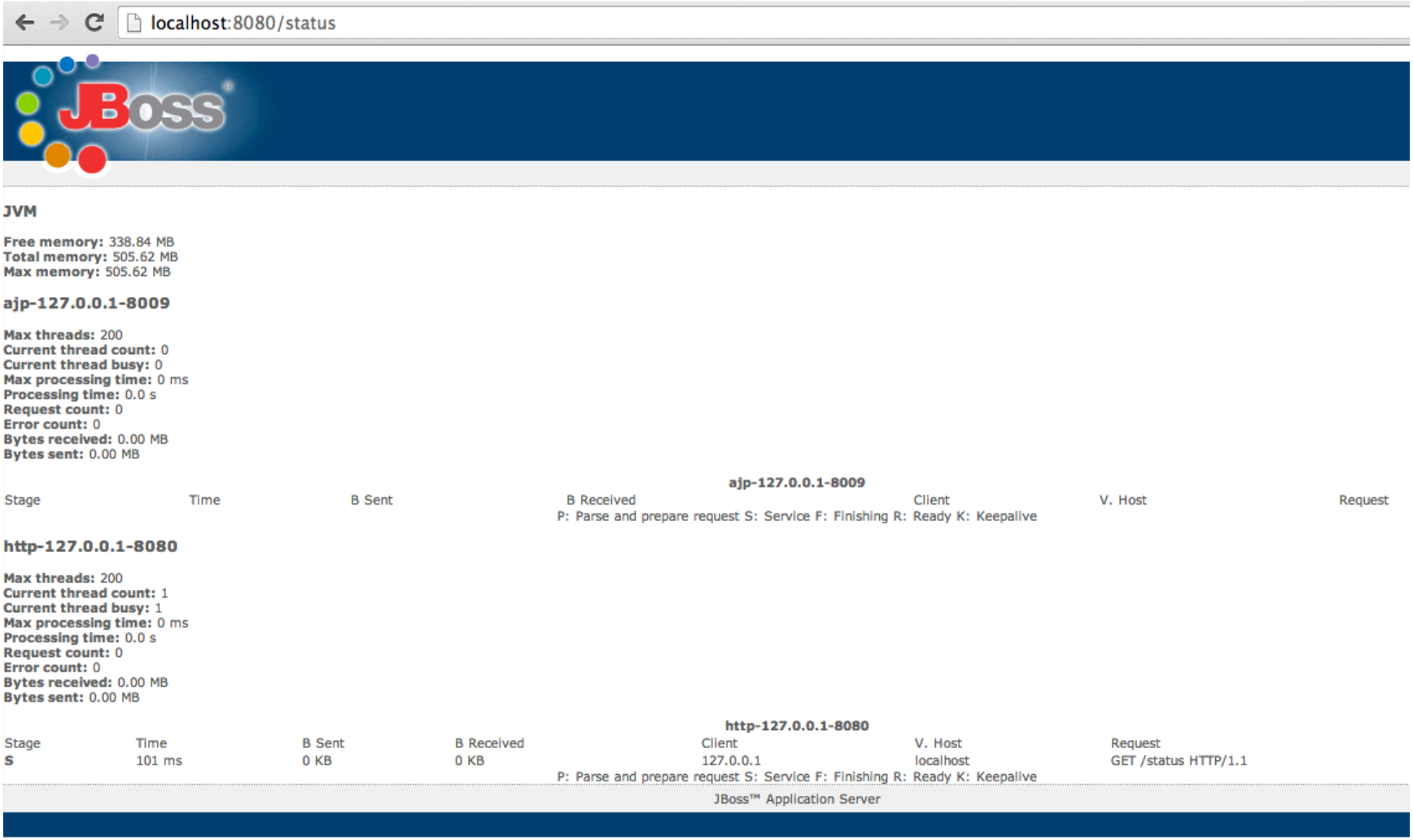
Unless you care to preserve compiled JSPs, this directory can be cleared (deleted) between JBoss restarts.

JBoss AS: Startup

- Su Unix/Linux/Mac OS X
  - eseguire via shell **$JBOSS_HOME/bin/run.sh**
- Su Windows
  - eseguire via shell **$JBOSS_HOME\bin\run.bat**
- Avvio di default su **localhost** (127.0.0.1)
- Possibile avvio come “system service”
  - il server viene lanciato all’avvio dell’host

JBoss AS: Startup

Se tutto va a buon fine su **http://localhost:8080/status**…



JBoss AS: Shutdown

- Su Unix/Linux/Mac OS X
  - **/etc/init.d/jboss stop** se il server è stato avviato come un serivzio di sistema
  - **$JBOSS_HOME/bin/shutdown.sh –S**
- Su Windows
  - **NET STOP JBoss** se il server è stato avviato come un serivzio di sistema
  - **$JBOSS_HOME\shutdown.bat**
- **Ctrl+C**
  - se il server è stato avviato in “foreground” usando lo script run

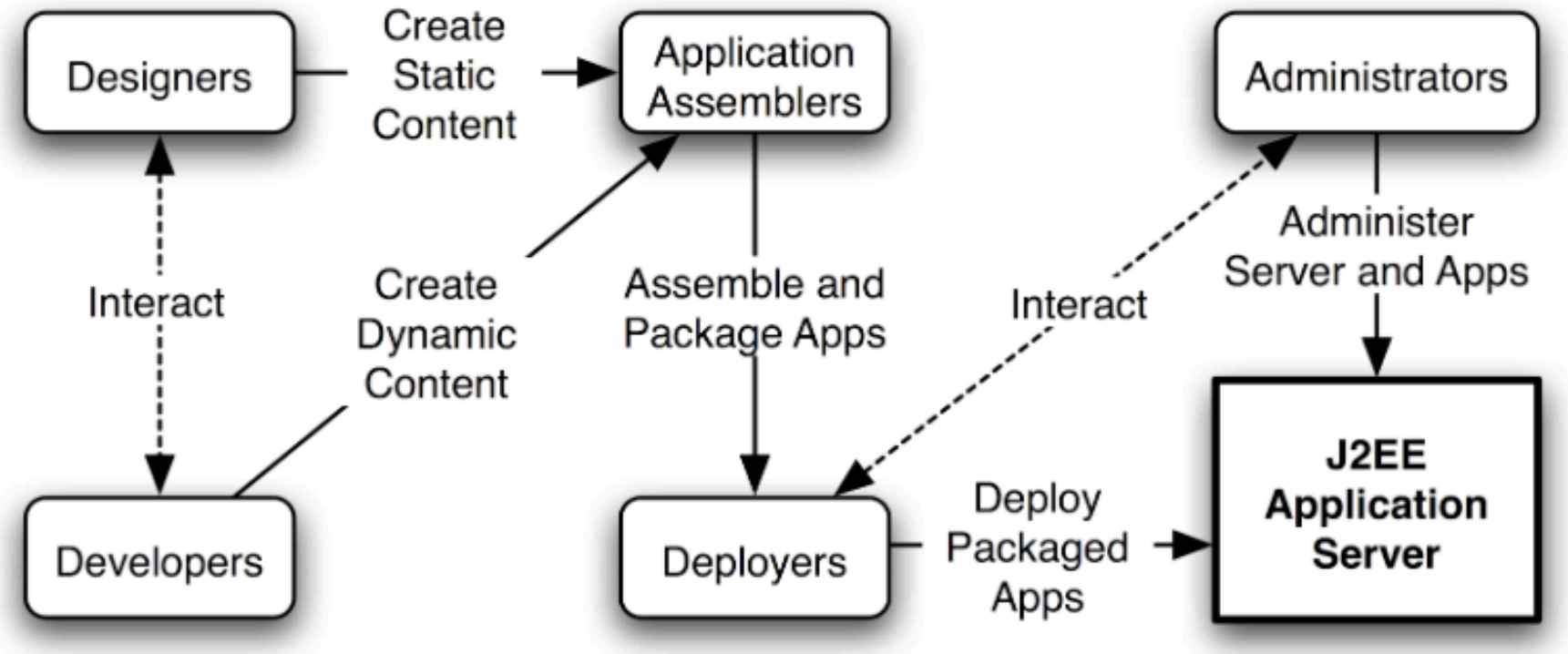
Eclipse + JBoss AS

- Ambiente che useremo durante il corso
- **JBoss AS** si integra con **Eclipse** tramite un set di plugins chiamati **JBoss Tools**
- L'integrazione consente il controllo dell'intero ciclo di sviluppo, deployement, debugging, monitoring di applicazioni Java EE
- Startup/Shutdown del server JBoss direttamente da interfaccia Eclipse
- ***NOTA:*** *consigliato mantenere la gestione di JBoss separata da Eclipse*

Ciclo di Vita Applicazioni Java EE

- Ciclo di sviluppo/deployment
  - Progettazione contenuti statici (HTML, CSS, etc.)
  - Sviluppo contenuti dinamici (Servlets, JSPs, EJBs, etc.)
  - Deployment descriptors (web.xml, application.xml, ejb-jar.xml, etc.)
  - Packaging (JAR, WAR, EAR, etc.)
  - Deployment packages (JAR, WAR, EAR, etc.) su Java EE server (JBoss AS)
  - Gestione applicazioni Java EE in esecuzione sul server

Ciclo di Vita Applicazioni Java EE



Deployment Descriptors

- Sono files che contengono le “istruzioni” per un dato container su come usare e gestire i componenti Java EE
  - Sicurezza
  - Transazioni
  - Persistenza
- Customizzabili (XML-based)
- Garantiscono la portabilità dei componenti

Deployment su JBoss AS

- Deployment (**2 fasi**):
  - Copia dell'applicazione (packaged) all'interno del server specificato da **${jboss.server.home.url}/deploy/**
  - JBoss si occupa di rendere l'applicazione pronta all'uso
- Undeployment (**2 fasi**):
  - Rimozione dell'applicazione (packaged) dal server specificato da **${jboss.server.home.url}/deploy/**
  - JBoss si occupa di disinstallare l'applicazione ed eliminare le sue risorse

Deployment su JBoss AS

- Supporto delle dipendenze necessarie
- **Hot** vs. **Cold** Deployment
- Le applicazioni packaged sono “scompattate” all’interno della directory **${jboss.server.temp.dir}/deploy/**
  - JBoss elimina il contenuto della directory ad ogni avvio
- Re-deployment automatico di tutti quei componenti i cui deployment descriptors vengono modificati mentre JBoss è in esecuzione
- Supporto di “nested deployments” (ad es. WAR all’interno di un EAR)

Deployers su JBoss AS

- Architettura di deployment “estendibile”
- Supporto nativo per:
  - JAR libraries
  - WARs
  - EARs
  - EJBs
  - Web Services
  - Client
  - …

Deployers su JBoss AS: WARs

- Web application ARchives (**.war**)
- Gestisce le applicazioni web che contengono i seguenti descrittori:
  - **WEB-INF/web.xml**
  - **WEB-INF/jboss-web.xml** (opzionale)
  - **WEB-INF/context.xml** (opzionale)

Deployers su JBoss AS: EARs

- Enterprise application Archives (**.ear**)
- Gestisce le applicazioni enterprise che contengono i seguenti descrittori:
  - **META-INF/application.xml**
  - **META-INF/jboss-app.xml** (opzionale)

Deployers su JBoss AS: EJBs

- Enterprise JavaBeans (**.jar**)
- Gestisce gli archivi di EJB che contengono i seguenti descrittori:
  - **META-INF/ejb-jar.xml**
  - **META-INF/jboss.xml** (opzionale)

Hot vs. Cold Deployment

- **Hot**
  - Veloce
  - Rischio di eccezioni sollevate dal ClassLoader
  - Proprietà di configurazione non riconosciute
  - Utile quando i cambiamenti riguardano JavaServer Pages che vengono comunque ricompilate dal Servlet engine
- **Cold**
  - Più lento ma stabile
  - Richiede lo shutdown di JBoss AS
  - Opzionalmente rimuove alcune dir temporanee
  - Re-deploy dell'applicazione da zero
  - Re-start di JBoss AS

Esercitazione: Deployment

- Deployment di un archivio web di esempio (**sample.war**)
- Avvio di JBoss AS
- Un-deployment (JBoss in esecuzione)
- Shutdown/Startup di JBoss
- Modifica del file **WEB-INF/web.xml**