



Delivering Excellence in Software Engineering



Contexts and Dependency Injection in Java EE 6

Olena Syrota

Java Lab Lead, RD Dep.

Olena_Syrota@epam.com

Agenda

- Architecture for distributed systems
 - Server objects
 - Server object containers
 - Services for server objects
- Java EE platform
 - What is missing in Java EE 5
- Inversion of Control (Ioc)
 - What is inverted
 - Sources of IoC
 - Dependency Injection
- Java EE CDI – IoC container
 - Examples
- CDI plus EJB

Server Objects

- Distributed enterprise applications:
 - Client-side architecture is easy
 - all of the mechanisms for scalability reside on the server side
- The most distinguished architecture in recent past is CORBA
 - In CORBA object is an entity with an object reference that provides the operations defined in its interface
 - In CORBA objects' implementations are managed on server, client gets remote reference to these objects



Server Object Lifecycle

- Server Object Lifecycle
 - Container manages server object lifecycle





- Java EE Server Objects
 - Servlet, JSF backing bean
 - EJB

- Java EE Containers
 - Web-container (Servlet, JSF backing bean)
 - EJB-container (EJB components)

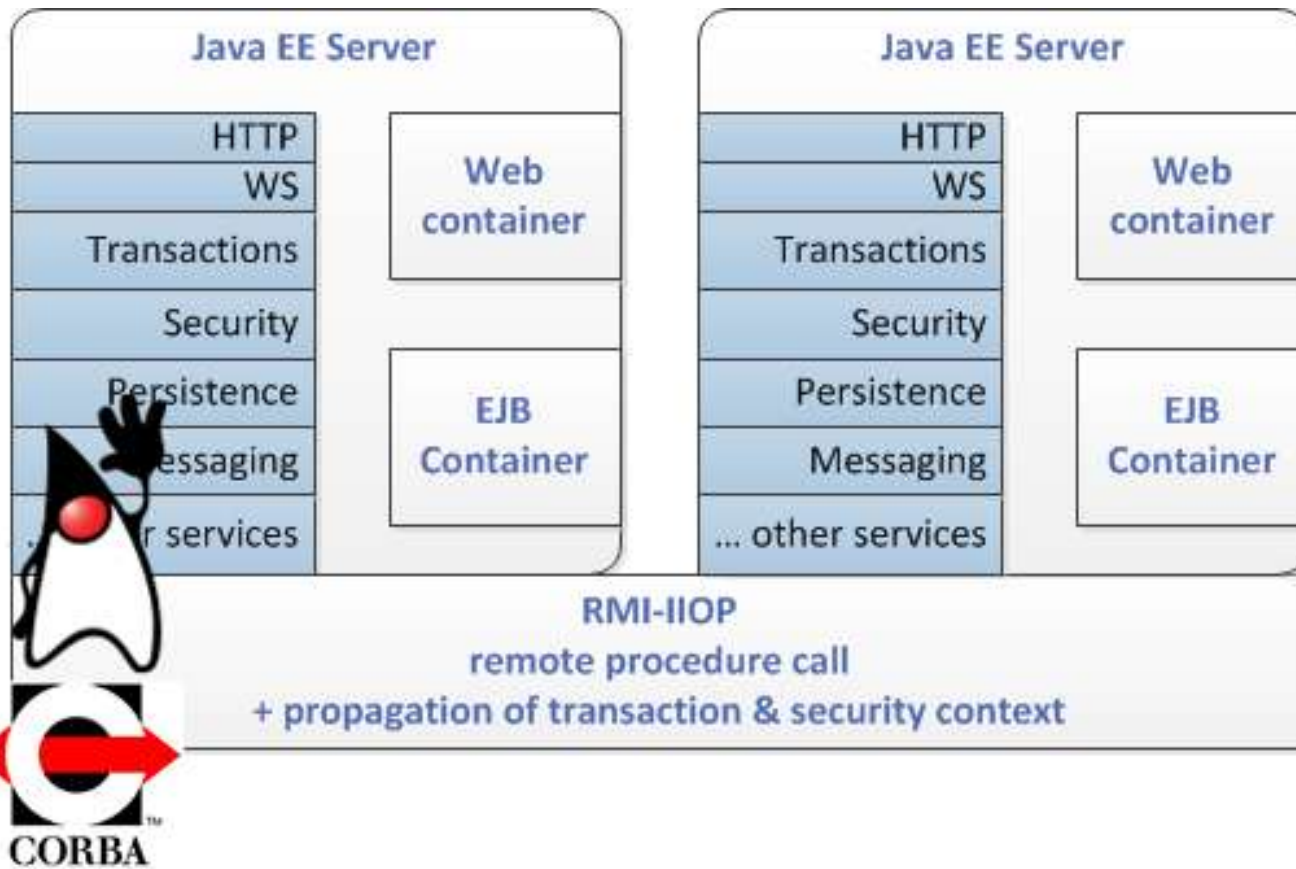
- How Java EE withstands high load?
 - Servlet
 - Singleton
 - EJB Stateless Session Beans
 - Pooling
 - EJB Stateful Session Beans
 - Passivate, activate

Services for Server Objects

- Important part of the CORBA standard
 - set of distributed services for distributed objects
- Services for distributed objects:
 - naming service
 - lifecycle management
 - security
 - transactions
 - event notification
 - concurrency control
 - ...



Java EE 5 Platform



- Containers
- Services
- Communication
 - RMI-IIOP
 - HTTP
 - WS
 - Messaging
- Critical cross-cutting services
 - Distributed transactions
 - Security

Java EE 5 Platform

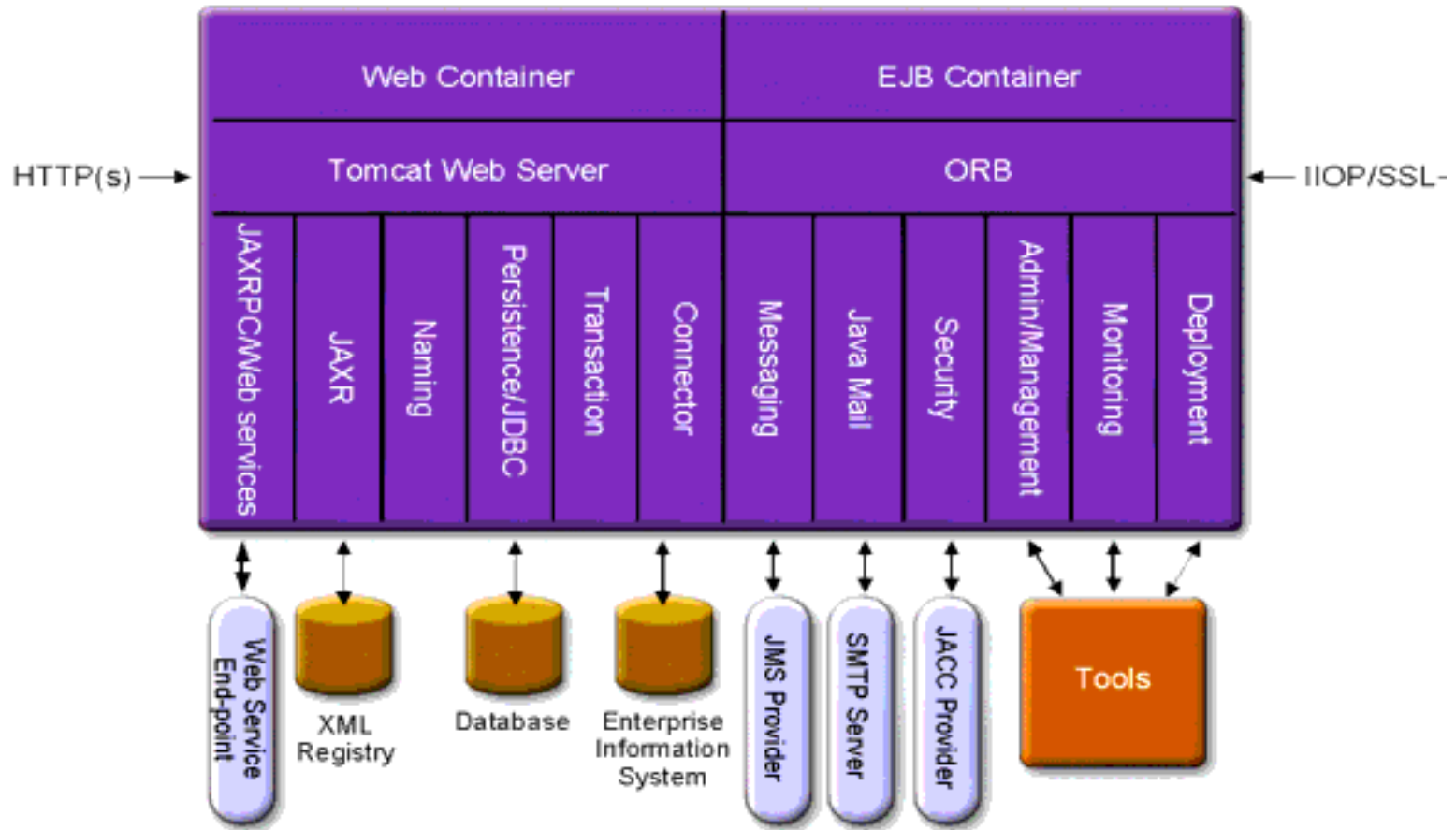
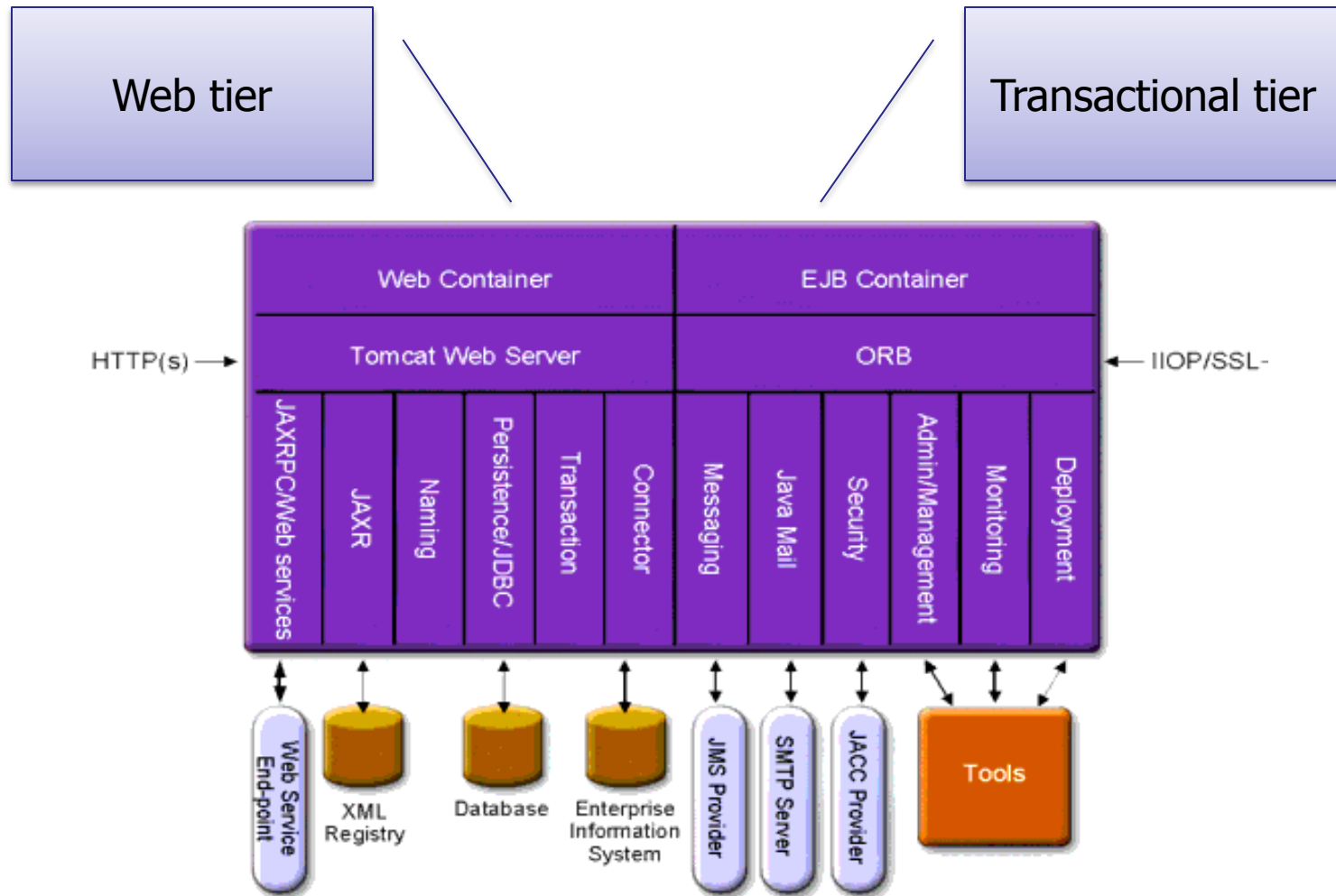


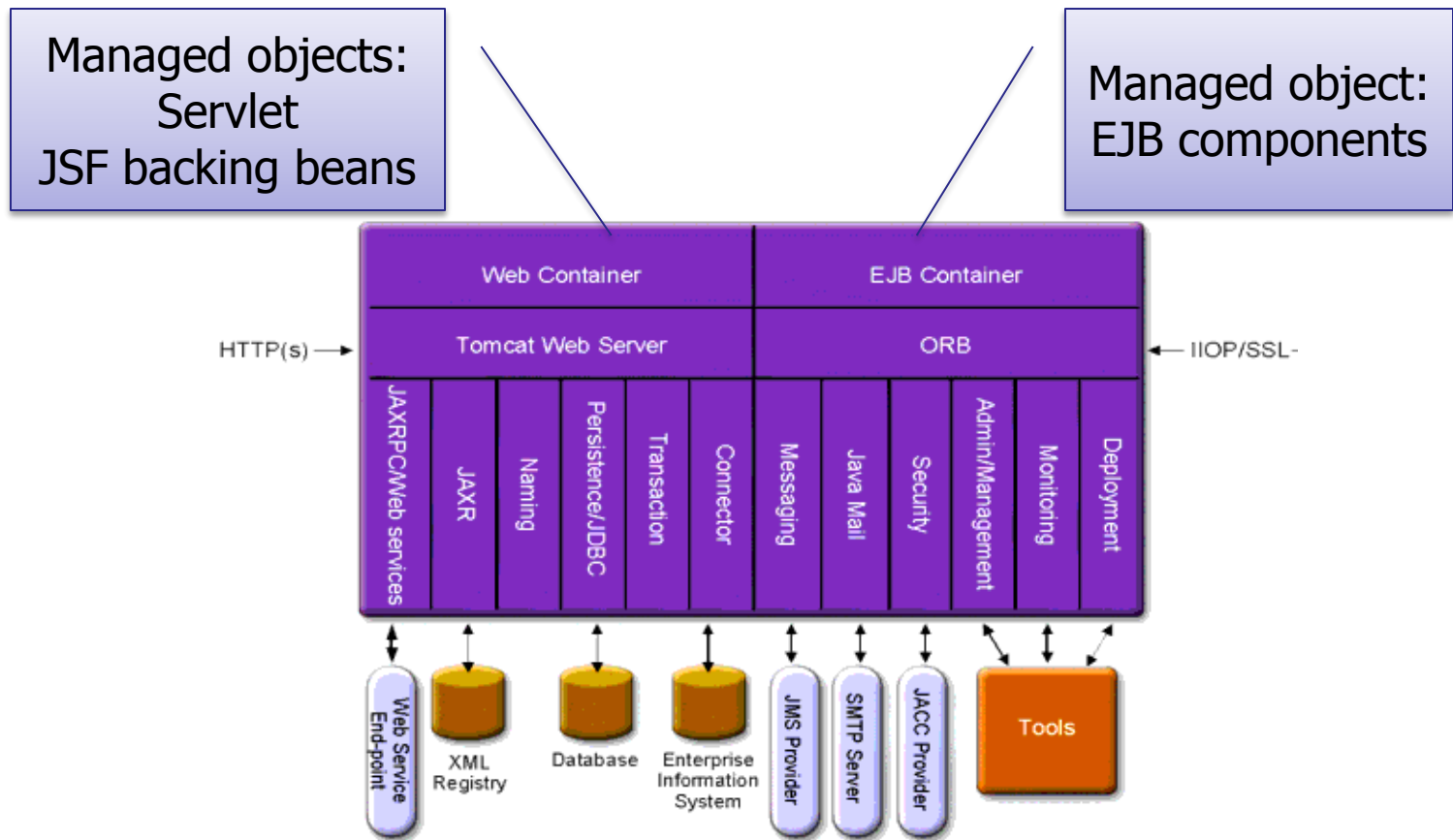
Figure: Sun Java System Application Server 9.1

Java EE tiers



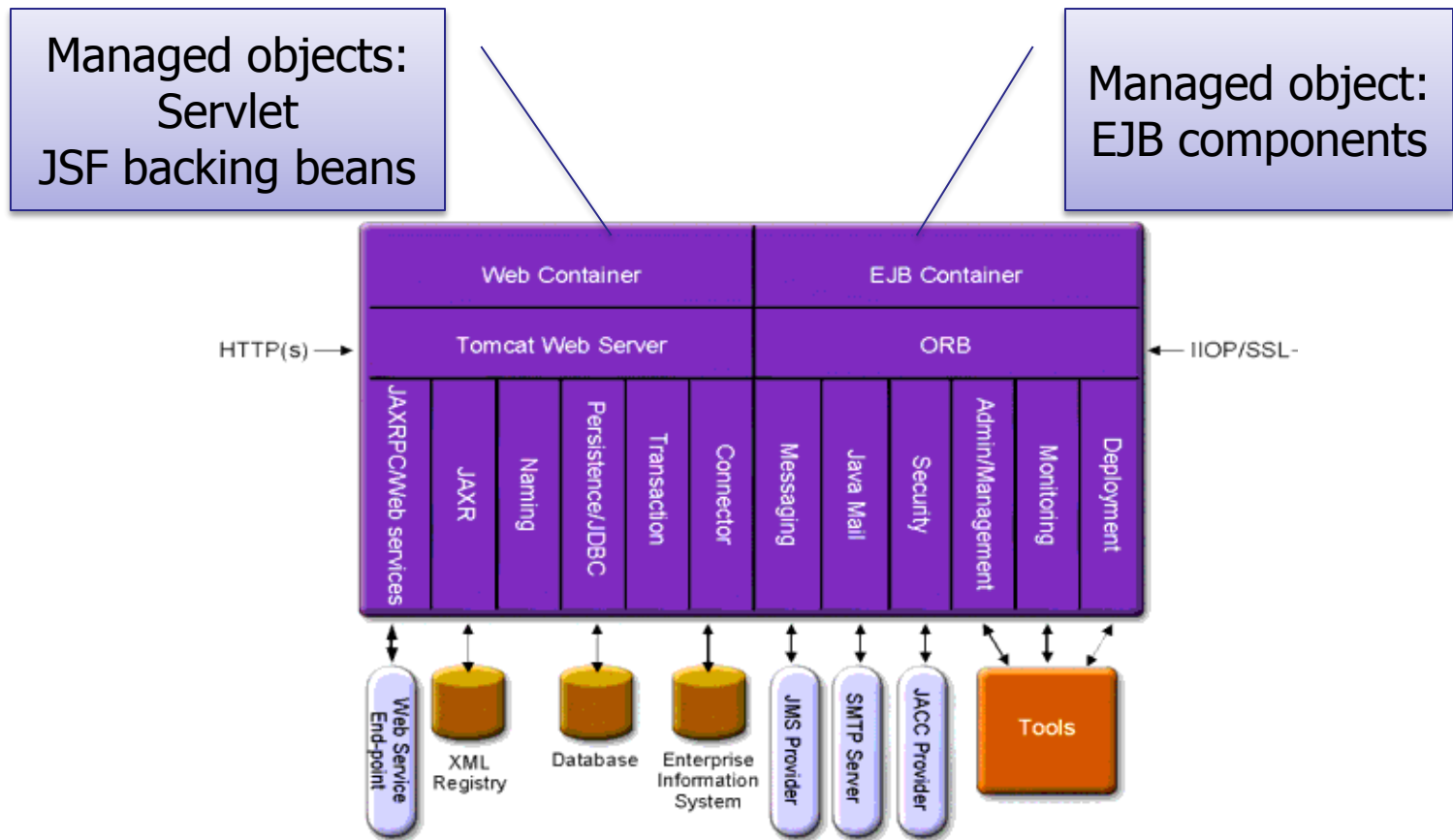
Predefined Set of Server Objects

- Java EE 5: predefined set of server objects managed by containers



Custom Server Object?

- How container can manage custom server object?
 - Custom server object like “command”, “service” etc.



What is missing in Java EE 5?

- Container for custom server objects is needed in Java EE
 - Inversion of Control (IoC) container
- But all advantages of Java EE 5 are still needed

Inversion of control

- Program control flow is inverted because container manages object lifecycle
- Object may be coupled to container:
 - It implements specific interfaces
 - It uses specific container's API to lookup dependent objects
- IoC helps to decouple object from container
 - No specific interfaces that object must implement
 - No calls of specific API to lookup dependent objects. **This is performed using dependency injection.**
- With **dependency injection** there is no explicit request for dependencies. Container wires dependencies.
- Hence **inversion of control**

Sources of “Inversion of control”

- Enterprise application needs enterprise architectures
 - Component architecture for server side logic

Component Architecture

Microsoft Application Architecture Guide v2. p. 24:

- “Components depend upon a mechanism within the platform that provides an environment in which they can execute, often referred to as **component architecture**.”
- Examples are the component object model (COM) and the distributed component object model (DCOM) in Windows; and Common Object Request Broker Architecture (CORBA) and Enterprise JavaBeans (EJB) on other platforms.
- Component architectures manage the mechanics of locating components and their interfaces, passing messages or commands between components, and—in some cases—maintaining state.”



Invasive component models

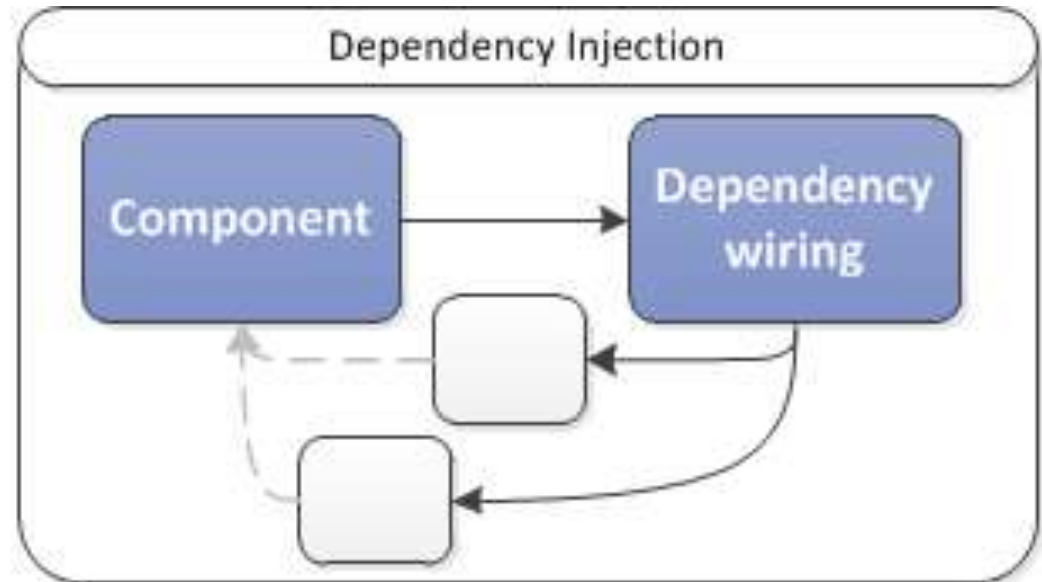
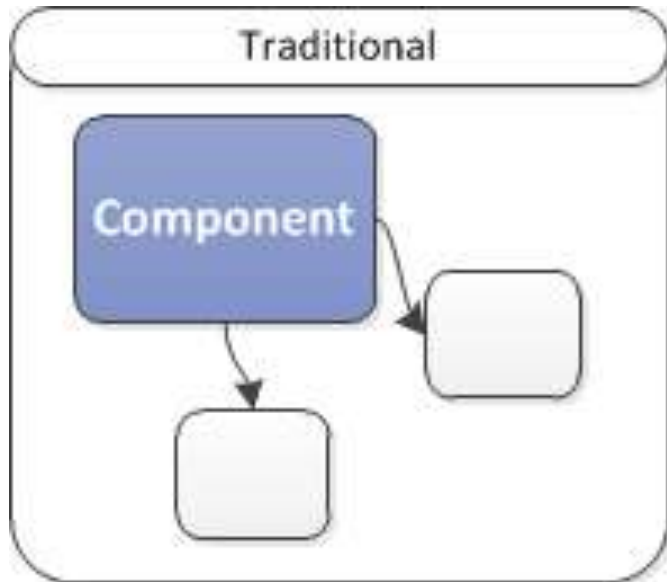
- Traditional component based development frameworks:
 - Based on **invasive component models**
 - Have **dependency lookup scenario**.
- **Invasive**
 - Developer should implement framework pre-defined interfaces
- **Dependency lookup scenario**
 - Component performs dependency lookup using framework specific APIs. Lookup in global registry.
- This leads to
 - Non-portable component programming model
 - Non-testable component. Components could only be tested and used within their originally intended frameworks
- **Examples: EJB, CCM (CORBA Component Model)**

Non-invasive component model

- IoC frameworks are
 - **non-invasive**
 - use the **dependency injection/setting scenario**.
- **Non-invasive**
 - component should not extend any framework specific interfaces
- **dependency injection/setting scenario**
 - This is an inverse version of the traditional *lookup/resolving* scenario
- **This decouples component from container**
- Examples: Spring, PicoContainer, Google Guice, Tapestry IoC, ...

Dependency Injection

- When container instantiate component, it assembles all its dependencies
- Dependency injection separates assembling from components



IoC Container Responsibilities

- Component lifecycle management
 - Statefull objects
- Component configuration
- Component dependencies wiring

Dependency Injection in Java EE 5

- Closed set of injectable resources provided by web-container and ejb-container
 - @EJB
 - @PersistenceContext, @PersistenceUnit
 - @Resource (Java EE Resource, e.g. DataSource, JMS destination; UserTransaction)

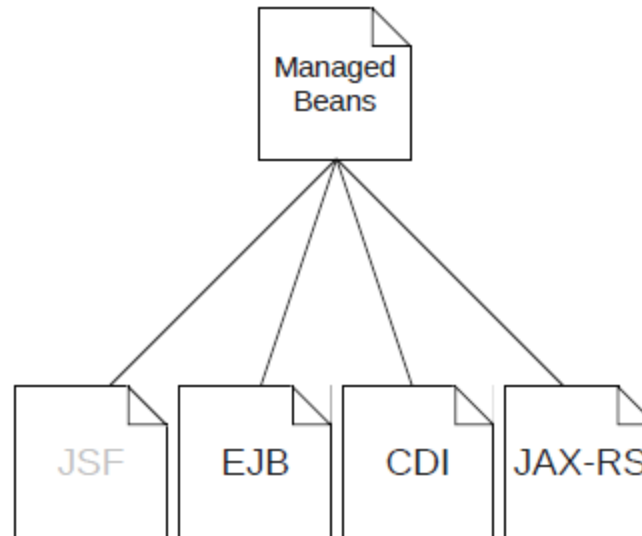
- CDI (Contexts and Dependency Injection) for Java EE Platform, JSR 299
 - IoC Container in Java EE 6
- CDI provides generic services applicable to all managed beans
 - Lifecycle management
 - Dependency injections
 - Event notification
 - Contexts
 - EL integration (bean names)

- Simplify the creation of applications that use both web tier and business tier technologies



Managed Beans in Java EE 6

- Java EE 6 platform define “managed bean” as container-managed objects
- Managed beans:
 - EJB session beans
 - annotated with `@ManagedBean`
 - objects with minimal programming restrictions, aka POJO (Plain Old Java Object). No special declaration/annotation required.



- CDI bean
 - Objects with minimal programming restrictions, aka POJO (Plain Old Java Object). No special declaration/annotation required.
- Most Java classes can be managed by CDI container

Examples

```
public class Login implements Serializable {  
  
    private Credentials credentials;  
  
    public boolean login() {  
        // login using credentials  
        ...  
    }  
}
```

```
public class Login implements Serializable {  
  
    @Inject  
    private Credentials credentials;  
  
    public boolean login() {  
        // login using credentials  
        ...  
    }  
}
```

- Where @Inject
 - Field, initializer method parameter (e.g. setter)

```
@Named("login")  
public class Login implements Serializable {  
  
    @Inject  
    private Credentials credentials;  
  
    public boolean login() {  
        // login using credentials  
        ...  
    }  
}
```

```
@Named("login") @SessionScoped
public class Login implements Serializable {

    @Inject
    private Credentials credentials;

    public boolean login() {
        // login using credentials
        ...
    }
}
```

```
<h:form>
  <h:panelGrid columns="2" rendered="#{!login.loggedIn}">
    <h:outputLabel for="usr">Username:</h:outputLabel>
    <h:inputText id="usr" value="#{login.credentials.username}"/>
    <h:outputLabel for="pwd">Password:</h:outputLabel>
    <h:inputText id="pwd" value="#{login.credentials.password}"/>
  </h:panelGrid>
  <h:commandButton value="Login"
    action="#{login.login}"
    rendered="#{!login.loggedIn}"/>
  <h:commandButton value="Logout"
    action="#{login.logout}"
    rendered="#{login.loggedIn}"/>
</h:form>
```

What can be injected

- What can be injected
 - Beans
 - EJB session bean
 - Resources (Java EE Resources, persistence contexts, persistence units, remote EJBs and web services)
- What else?
 - “Produced” beans

Producer Methods

```
@Named @SessionScoped
public class Login implements Serializable {

    @Inject private Credentials credentials;
    private Client client;

    public void login() { ... }
    public void logout() { ... }

    @Produces @LoggedIn
    public Client getLoggedInClient() {
        return client;
    }
}
```

@LoggedIn – qualifier interface that assign semantic meaning

Client of Producer Method

```
@Named @RequestScoped  
public class SomeController {  
  
    @Inject @LoggedIn Client client;  
    ...  
}
```

Produce Random Numbers Example

```
@ApplicationScoped
public class RandomNumberGenerator {

    private Random random=new Random(System.currentTimeMillis());

    @Produces @Named @Random
    int getRandomNumber() {
        return random.nextInt(100);
    }
}
```

Event Notification (1/3)

```
@Named @SessionScoped
public class Login implements Serializable {

    private Client client;
    @Inject @LoggedIn private Event<Client> userLoggedInEvent;
    @Inject @LoggedOut private Event<Client> userLoggedOutEvent;

    public void login() {
        client = ...
        if (client!=null)
            userLoggedInEvent.fire(client);
    }

    public void logout() {
        if (client!=null) {
            userLoggedOutEvent.fire(client);
        }
        client = null;
    }
}
```

Event Notification (2/3)

```
@Qualifier  
@Retention(RUNTIME)  
@Target({METHOD, FIELD, PARAMETER, TYPE})  
public @interface LoggedIn {}
```



qualifier

```
@Qualifier  
@Retention(RUNTIME)  
@Target({METHOD, FIELD, PARAMETER, TYPE})  
public @interface LoggedOut {}
```

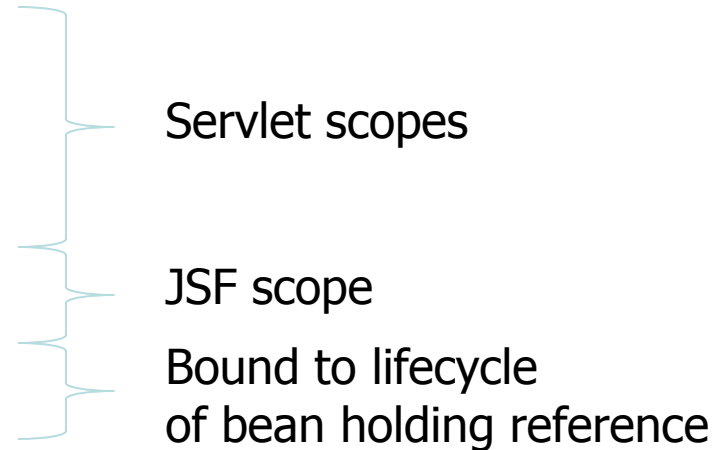
Event Notification (3/3)

```
public class DoSomeLogicOnLogin {  
  
    void onLogin(@Observes @LoggedIn Client client) {  
        ...  
    }  
  
    void onLogout(@Observes @LoggedOut Client client) {  
        ...  
    }  
}
```

- Java EE (servlets, EJBs) do not have a well-defined *scope*

- CDI defines built-in scopes:

- @RequestScoped
- @SessionScoped
- @ApplicationScoped
- @ConversationScoped
- @Dependant



- Web + Transactional:

- EJB Stateful beans can now be bound to context have one of above scope

CDI plus EJB

- EJB provides advanced enterprise services
- CDI provides IoC Container capabilities
 - doesn't provide any transactional, monitoring, or concurrency aspect out of the box
- EJB + CDI = synergy of both technologies

- When should you use a session bean instead of a plain managed bean? Whenever you need:
 - method-level transaction management and security
 - concurrency management
 - instance-level passivation for stateful session beans and instance-pooling for stateless session beans
 - remote or web service invocation
 - timers and asynchronous methods
 - JMX monitoring
- If you do not need these features, use ordinary managed bean

Concurrent access to CDI beans

- @SessionScoped and @ApplicationScoped beans are available for concurrent access
 - For this concurrency management provided by EJB 3.1 is useful
 - Consider session and application scoped beans to be EJBs

Well defined scope for EJBs

- Stateful EJBs now can be bound to well-defined contexts
 - session, request, application, conversation

- **In Java EE 6, the "EJB 3.1 with CDI" combination is the perfect synergy**

- **Weld**
 - GlassFish 3.0
 - Jboss 6.0.0
 - Tomcat 6.0.18 or later
 - Jetty 6.1.x

Bibliography

- ORB Basics. http://www.omg.org/gettingstarted/orb_basics.htm
- **Ke Jin.** Why and what of Inversion of Control. <http://www.pocomatic.com/docs/whitepapers/ioc/>
- **Martin Fowler.** Inversion of Control Containers and the Dependency Injection pattern. - <http://martinfowler.com/articles/injection.html#InversionOfControl>
- **Dan Allen.** Moving to Java EE 6 and CDI and away from the clutter. - <http://www.slideshare.net/mojavelinux/javaaee6-andcdireduceclutter>
- JSR-299: The new Java standard for dependency injection and contextual lifecycle management. <http://docs.jboss.org/weld/reference/latest/en-US/html/index.html>
- **Adam Bien.** Enterprise JavaBeans 3.1 with Contexts and Dependency Injection: The Perfect Synergy. - <http://www.oracle.com/technetwork/articles/java/ejb-3-1-175064.html>
- Weld - JSR-299 Reference Implementation. - <http://docs.jboss.org/weld/reference/1.0.0/en-US/html>



Delivering Excellence in Software Engineering

Contexts and Dependency Injection in Java EE 6



For more information, please contact:

Olena Syrota, PhD in information technology

Java Lab Lead

EPAM Systems, Inc.

Olena_Syrota@epam.com

<http://www.epam.com>