From J2EE to Java EE

Filippo Diotalevi

http://www.diotalevi.com
filippo.diotalevi@jugmilano.it

November 23rd, 2006
Summary

“... from J2ee to S+H, and back to Java EE..”

Java EE reference architecture?
Core J2ee Patterns - 2001-2003

The 'reference architecture' for J2EE applications

Collection of patterns for the development of enterprise applications
Did you say... Patterns?
Did you say... patterns?
Core J2ee Patterns – Web Tier

View Controller → Front Controller

Front Controller → Context Object

Context Object → Action Controller

Action Controller → Intercepting Filters

Intercepting Filters → Client

Client

Taken from “Pattern per lo sviluppo di applicazioni J2ee”, Filippo Diotalevi, Webbit 2004
Core J2ee Patterns – Business & Int. Tier

Taken from “Pattern per lo sviluppo di applicazioni J2ee”, Filippo Diotallevi, Webbit 2004

From J2EE to Java EE Tour – Jug Prague
Core J2ee Patterns - Technology mappings

- Service delegates → POJO
- Session Facades → Session Stateless Bean
- Services → Session Stateless Bean
- Persistence → Entity Beans
- Service Locators → JNDI
Core J2ee Patterns – Why?

- **Business Delegate**: reduce coupling between presentation-tier and business services

- **Session Facade**: encapsulate the complexity of interactions between the business objects participating in a workflow. The S.F. provides a uniform coarse-grained service access layer to clients

- **ServiceLocator**: abstract all JNDI usage and to hide the complexities of initial context creation, EJB home object lookup, and EJB object re-creation
Core J2ee Patterns – Pros & Cons
Pros:

- Standard based
- Decoupled architecture
- Easy to split work between team members
- Highly documented
Core J2ee Patterns - Pros & Cons
Core J2ee Patterns - Pros & Cons

Cons:

- Complex
- Verbouse
- Can be inefficient
- Complex
- Did I say,, complex?
2004-2006: “J2EE development without EJB”?!?!
2004-2006: “J2EE development without EJB”?!!

**Powerful JavaBeans-based configuration management, applying Inversion-of-Control (IoC) principles**

.. an object-relational mapping (ORM) solution for the Java language...

Mix together

1/3 - Your favourite web framework
1/3 - Spring Framework
1/3 - Hibernate

Simple, efficient and flexible architecture
Spring+Hibernate based architectures

Your favourite Web Framework here (es. Struts)

Service Layer

Integration Layer

Dependency injection

DAO

DAO
public class PersonHibernateDAO extends HibernateDaoSupport implements PersonDAO {

    public List listDocuments() {
        return this.getHibernateTemplate().
            .find("from com.myapp.domain.Document order by code");
    }

    public void deletePerson(Person person) {
        this.getHibernateTemplate().delete(person);
    }

    ....
}

public class TaxServiceImpl implements TaxService {
    private PersonDAO personDAO;
    public PersonDAO getPersonDAO() {
        return personDAO;
    }
    public void setPersonDAO(PersonDAO personDAO) {
        this.personDAO = personDAO;
    }

    public List findStudents(TaxStudentQueryParams params) {
        ....
Dependy injection in action

```xml
<bean id="personDAO" class="com.pnetx.pulse.dao.hibernate.PersonHibernateDAO">
  <property name="sessionFactory">
    <ref bean="sessionFactory"/>
  </property>
</bean>

<bean id="taxServiceTarget" class="com.pnetx.pulse.spring.services.TaxServiceImpl">
  <property name="taxDAO">
    <ref bean="taxDAO"/>
  </property>
  <property name="personDAO">
    <ref bean="personDAO"/>
  </property>
  <property name="academicSemesterDAO">
    <ref bean="academicSemesterDAO"/>
  </property>
  <property name="userManager">
    <ref bean="userManager"/>
  </property>
</bean>
```
S+H Architecture - Pros & Cons
Pros:

- Really decoupled architecture
- Lightweight (less patterns, less infrastructure)
- Really powerful
- Simple (see projects Appfuse or Equinox)
S+H Architecture – Pros & Cons
Cons:

- Non-standard
- Less documented
Java EE 5
Crash Course
Java EE 5 released on May 2006

Mantra: “EOD – Easy of Development”
Annotations
Resource injection
EJB 3:

- POJO Implementation
- POJI Business Interface
- Annotations
- “Configuration by exception”
- Java Persistence API
- No more Home interfaces!
EJB 3:

@Stateless
@Stateful
@Entity
@MessageDriven
Java Persistence API:

- EntityManager API
- EJB QL
- Query API
JavaEE5 Crash Course - 6'31''

- Java Persistence API (JSR-220)
- EJB 3.0 (JSR-220)
- JavaServer Faces 1.2 (JSR-252)
- JSP 2.1 (JSR-245)
- JAX-WS 2.0 (JSR-224)
- JAXB 2.0 (JSR-222)
- Common Annotations (JSR-250)
- StAX (JSR-173)
Java EE 5 Architecture

- Inspired by S+H Architecture
- Uses EJB 3.0 and dependency injection
- All standard based
A reference architecture for Java EE 5

Put your favourite web framework here

Delegate → Service → BO → Data Access Object → DBA

Delegate → Service → BO → Data Access Object

Delegate → Service

Stateless Session Bean

Entity

Session Bean

Stateless

JPA

Dependency injection

NEW!
@Stateless

public class PageManagerImplementation implements PageManager {

@EJB public PageDao dao;

public void save(Page pagina) {
    pagina.setLastModified(new Date());
    dao.save(pagina);
}

@Local

public interface PageManager {

public void save(Page pagina);

//..
}

//..
```java
@Entity
public class Page implements Serializable{
    private String title, content;
    private Date lastModified;

    @Id public String getTitle() { return title; }

    @Temporal(TemporalType.DATE)
    public Date getLastModified() { return lastModified; }

    public String getContent() { return content; }

    //... setters and other methods follow...
}
```
@Stateless

public class PageDaoImplementation implements PageDao {
    @PersistenceContext
    EntityManager entityManager;
    public void save(Page pagina) {
        entityManager.merge(pagina);
    }

    public interface PaginaDao {
        public void salva(Pagina pagina);
    }

    public interface PaginaDao {
        public void salva(Pagina pagina);
    }
}
Java EE 5 dependency injection is less powerful and flexible than Spring d.i.

Dependency injection can only be used by managed classes

Use Spring to manage EJB services and inject them in the front end

@Questions?

Thank you!
References

• JSR 220: Enterprise JavaBeansTM 3.0

• Glassfish website and documentation
  https://glassfish.dev.java.net

• JBoss website and documentation
  http://www.jboss.org

• Gavin King's “Entity Beans in EJB 3”

• Kenneth Saks “Enterprise Java Bean 3.0”

• Filippo Diotalevi's “Java EE 5 - Step by Step”
  http://www.diotalevi.com/weblog/?page_id=125
Filippo Diotalevi

http://www.diotalevi.com
filippo.diotalevi@jugmilano.it