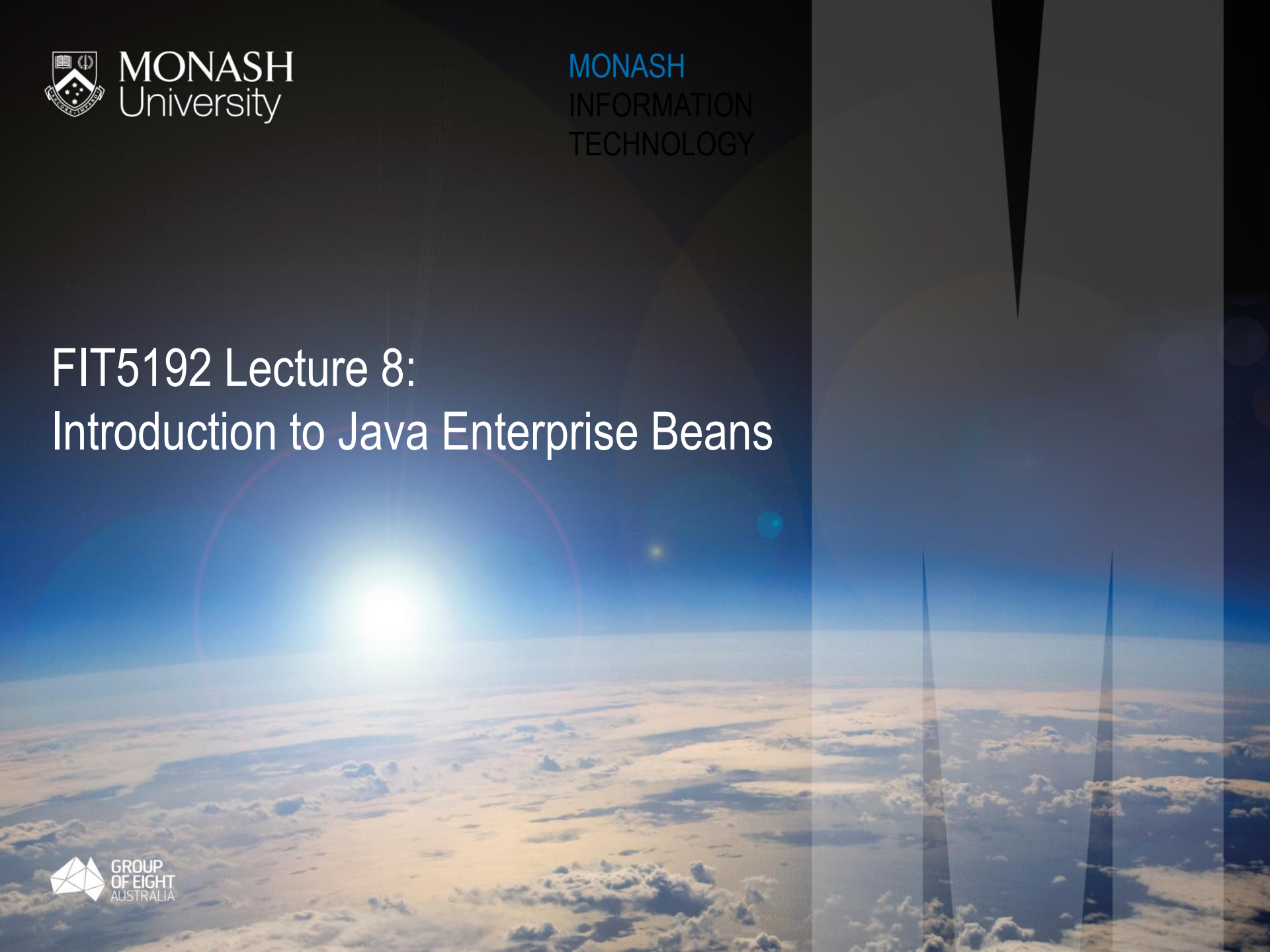

MONASH University
MONASH
INFORMATION
TECHNOLOGY
FIT5192 Lecture 8:
Introduction to Java Enterprise Beans
GROUP
OF EIGHT
AUSTRALIA

MONASH University

Last Lecture

- More advanced ORM
- Criteria API
- Container Managed Entity Manager

GROUP OF EIGHT AUSTRALIA

MONASH University

This Lecture

- Provide an understanding of the role Enterprise Java Beans has in developing enterprise web applications with the Java EE platform.
- Review the role of Session Beans in implementing business logic code.

GROUP OF EIGHT AUSTRALIA

Enterprise JavaBeans

What are Enterprise JavaBeans (EJBs)?

- **Java EE components that implement the business logic for enterprise applications.**
  - Run in an EJB container which operates in a runtime environment such as the Glassfish server.
- Similar to ManagedBeans, EJBs have their own specific annotations to designate what kind of functionality should be implemented.
- **Designed to access databases, be called/invoked by local and remote clients or components**

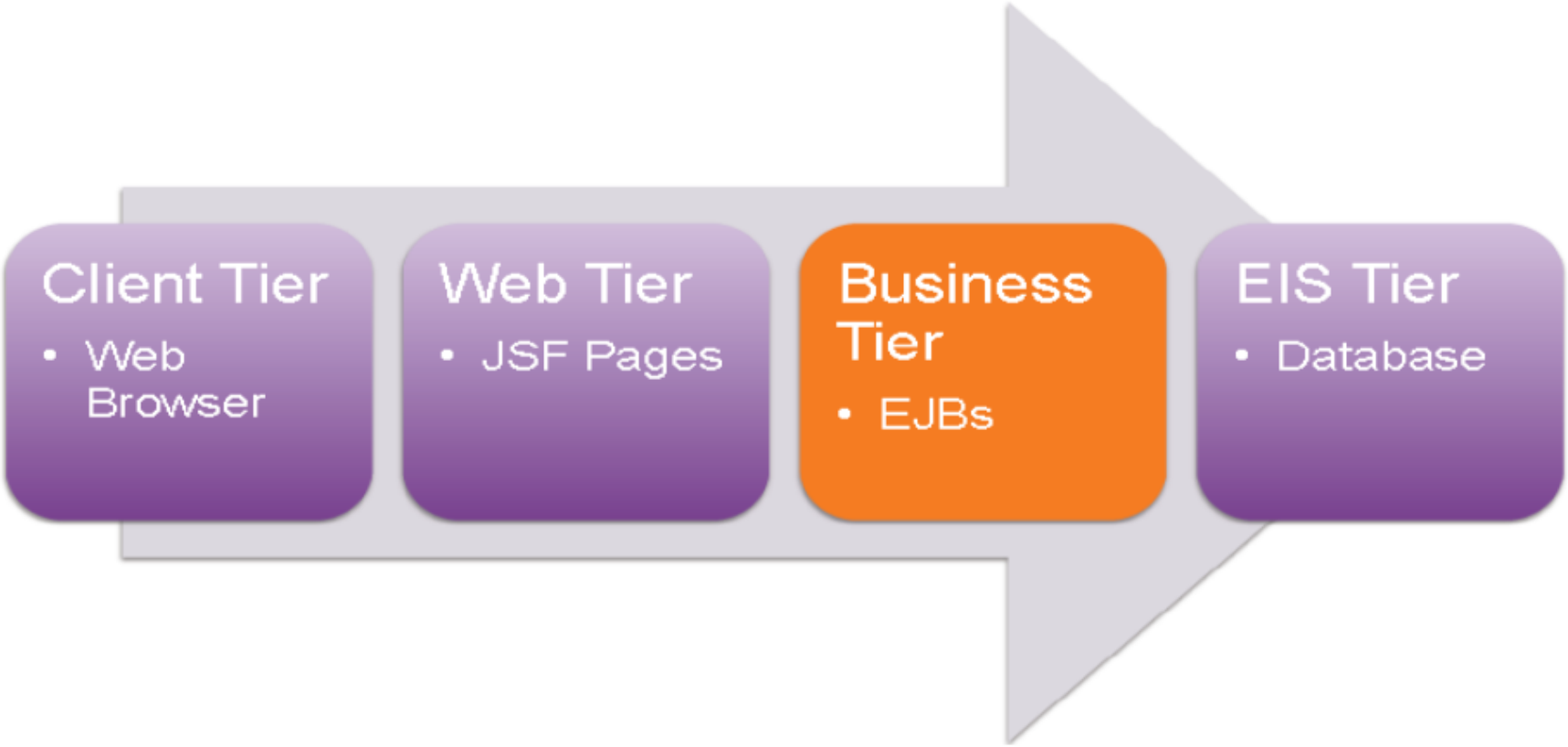
Benefits of EJBs

- **Simplify development of large-scale enterprise applications by separating Business-tier from Web-tier.**
  - Client developers can focus on the web frontend.
  - Application developers can focus on the web backend.

- **EJB container has access to system-level services such as transactions and security authentication.**
  - Enables developers to focus on the core business application objectives.

- **EJBs are also portable components which can be reused in other Java EE applications provided the standard Java libraries have been used.**

Where do EJBs belong?



Types of Enterprise Beans

- **EJBs can be split into two key types:**
  - **Session Bean (Focus for this Lecture):** Performs a task relating to the business logic of the application.
  - **Message-driven Bean:** Listens for specific messaging types and acts accordingly. Based around the Java Message Service API.

Session Bean

- **Concerned with the business logic of the enterprise application.**
  - Hides the complex interactions taking place from the client and enables access with database and transactional services.
  - Accessible via local, remote or web service client method invocation.
- **Three key types of Session Beans:**
  - Stateless: @Stateless
  - Stateful: @Stateful
  - Singleton: @Singleton

Stateless Session Beans

- The most popular session bean component within Java EE applications.
- **Simple yet powerful approach in responding to business operations.**
  - Stateless in this context means completing an operation within a single method call.
  - Client state is not retained once a method call has been completed.
- **Easily scales and supports multiple clients via a pool of instantiated beans.**
- Possible to implement web services using this bean.

Stateful Session Beans

- **Concerned with preserving the conversational state between components.**
  - Unable to be shared; assigned to a single client.
  - Used for tasks which have to be done in multiple steps.
  - Can be viewed as a Bean which has access to majority of different scopes.

- **One-to-one relationship with the client during the conversation process.**
  - Can be memory intensive should there be many clients.
  - Terminated instance once the client has removed the bean.

Singleton Session Beans

- **Session bean which is only instantiated once per application instance.**
  - Comparable to @ApplicationScope annotation used in Managed Beans.
  - Useful for when global access to the EJB is required.

- **A common implementation for Singleton session beans are using them for techniques such as *caching* data.**

Example: Simple EJB Structure

```
import javax.ejb.Stateless;
import javax.ejb.LocalBean;

@Stateless
@LocalBean
public class HelloWorldBean {
    String message = "Hello World!";
    public String sayHello(){
        return message;
    }
}
```

- We can write EJBs just like normal **POJOs** but we use **annotations** instead to configure them.

Using Enterprise JavaBeans

- **We can access EJBs using a *no-interface view* or through a defined *business interface*.**
  - *No-interface views:* Expose the public methods of an EJB implementation to clients.
  - *Business Interface:* A standard Java Interface which outlines the business methods of the EJB.
- **Aside from the public method implementations, all other EJB method implementations or settings are hidden from the client.**
- **We can obtain a reference to an EJB via *dependency injection* (more in-depth in next lecture) or *JNDI* (Java Naming Directory Interface).**

Local and Remote Clients

- When designing EJBs, we also need to take into consideration how we will be using them in our application.
- **Must define business interfaces (if applicable) via @Local and @Remote respectively.**
- **Local Clients**
  - Must run in the same application that it accesses the EJB
  - Potentially higher performance due to no remote calls
- **Remote Clients**
  - Able to handle processing on different machines (scalable)
  - Business interface ***must*** *be used within the application.*

Accessing EJBs via Dependency Injection

- Working with EJBs is best done using dependency injection since it is very easy to implement.
- **To obtain a reference to the *no-interface view* or local business interface of an EJB, we use dependency injection via the *javax.ejb.EJB* annotation and specify the EJB implementation class.**

```
@EJB
CalculatorBean calculator;
```

- **Interfaces need to have a unique name separate from the Bean.**
  – **E.g.** Calculator**,** CalculatorLocal**,** CalculatorRemote**.**

Accessing EJBs via Dependency Injection

Accessing EJBs via JNDI (1)

Three key namespaces used for lookups:

java:global

- java:global[/app]/module/ejbName[/interfaceName]
- Portable way of finding remote EJBs (and when annotations aren't usable such as older Java EE platforms).

java:module

- java:module/ejbName[/interfaceName]
- Used to search for EJB implementations within the same EJB module.

java:app

- java:app[/module]/ejbName[/interfaceName]
- Used to search for EJB implementations within the same application.

Accessing EJBs via JNDI (2)

- Example: *CalculatorBean* inside *Calculator.war*
  - **Portable JNDI Name:**

```
java:module/CalculatorBean
```

OR

```
java:global/calculator/CalculatorBean
```

  - **Usage:**

```
Context context = new InitialContext();
CalculatorBean calc = (CalculatorBean)
context.lookup("CalculatorBean/sum");
```

- **We will be focusing on the Injection approach for this unit.**

EJB Naming Conventions

- **Oracle recommends the use of EJBs following a standard naming covention:**
  - Enterprise bean names and classes should follow the format of *nameBean.*
    - *E.g. CalculatorBean, ShoppingCartBean*
  - Business interfaces should consist of just the *name.*
    - E.g. Calculator, ShoppingCart
- **Reviewing many books or programming examples will show that many programmers don't always follow this pattern.**
  - Any naming convention that's consistent works best.

Contents of an EJB

- **EJBs are comprised of the following files:**
  – **Enterprise Bean Class:** Implements the business methods of the enterprise bean and any lifecycle callback methods.
  – **Business Interfaces:** Define the business methods implemented by the enterprise bean class. Interface is not required if the enterprise bean exposes a local, no-interface view.
  – **Helper Classes:** Other classes needed by the enterprise bean class, such as exception and utility classes.

Packaging EJBs

- EJBs can be packaged within an EJB JAR (Java Archive) file.
- **One or more EJBs JAR files and other modules can be packaged into an EAR (Enterprise Archive) file.**
  - This represents the EJB “module” that are used in NetBeans IDE.
- EJBs can also be packaged into a WAR (Web Application Archive) file should they implement a part of the application

Summary

- Provide an understanding of the role Enterprise JavaBeans has in developing enterprise web applications with the Java EE platform.
- Review the role of Session Beans in implementing business logic code.

Next Lecture

- More in-depth analysis of **Enterprise JavaBeans** and how we can use them to build more complex enterprise applications.

MONASH University

See you in the Studio !

Readings

MONASH University

- **Please review Chapter 26 *Enterprise* Beans in the Oracle Java EE 7 tutorial.**

Many good examples and breakdowns of key terminology.