

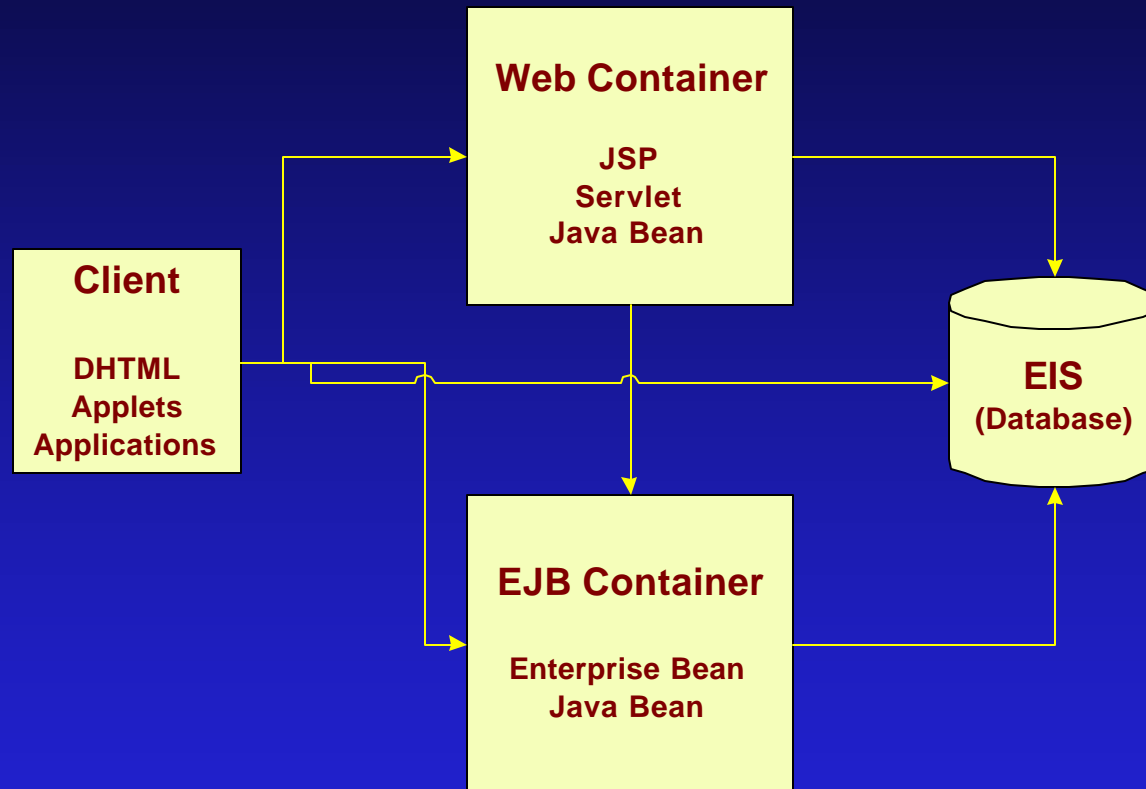


# **Architecture Alternatives in J2EE**

## **Benefits and Drawbacks**

Scott Wheeler  
Nortoc Inc.

# J2EE Overview



# Client - Presentation Layer

**Purpose:** To allow independence from the WebContainer (data provision) and EJB Container (Business Logic layer).

- } **Dynamic HTML** (via JSP, Servlet)
- } **Java Applet** (Swing, Java Plugin)
- } **Java Application** (Swing)

# Web Container – Data Provision

**Purpose:** To allow independence from the Client (presentation) and EJB Container (Business Logic layer).

- } **Dynamic HTML**
  - } JSP
  - } Servlet
- } **Data Pump – (XML, CSV, Serialized Object)**
  - } Servlet
  - } JSP
  - } Data Service

# EJB Container - Business Logic

**Purpose:** To centralize Business Logic to increase reuse and consistency and to allow independence from the Client (presentation) and Web Container.

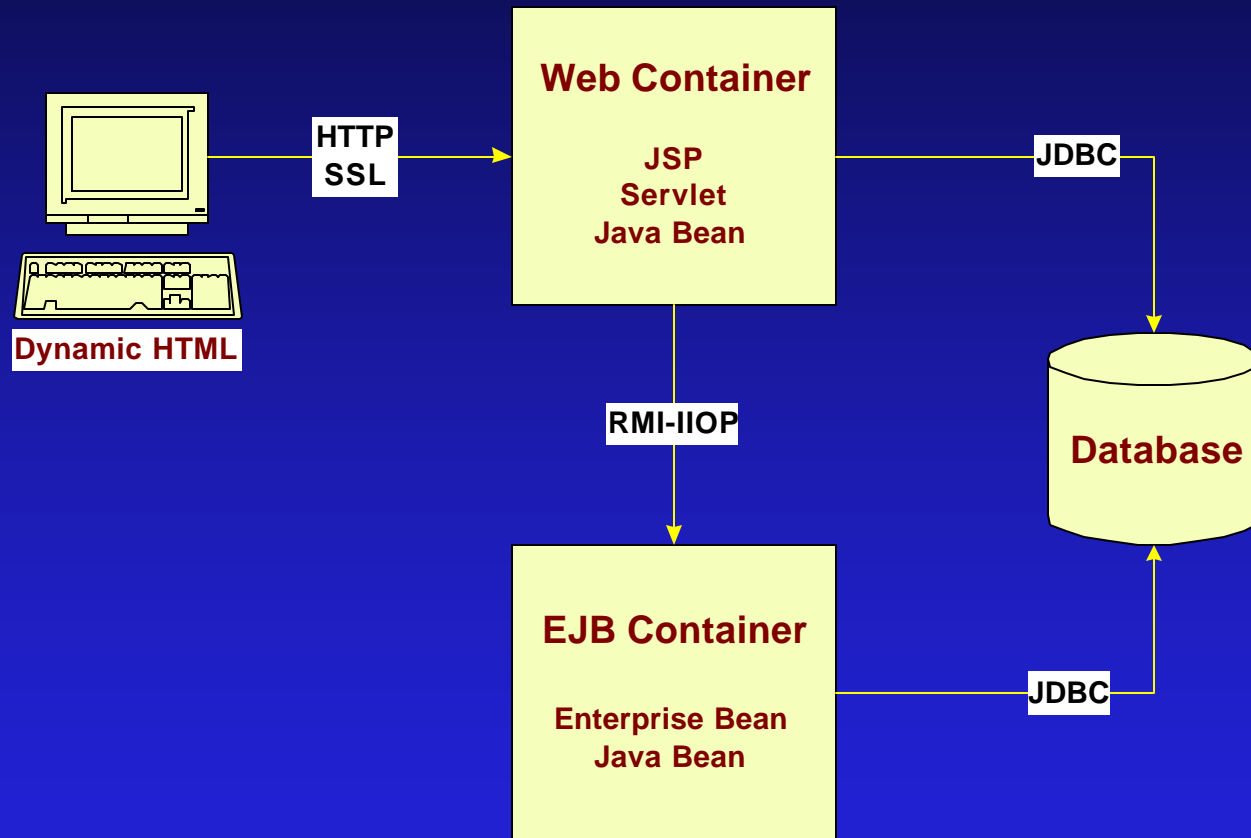
- } Entity Beans
- } Session Beans, Entity Beans – Session Beans provide the remote interface to Entity Beans
- } Session Beans Only
- } Session Beans, Application Classes - Session Beans provide the remote interface to Entity Beans

# EIS - Database

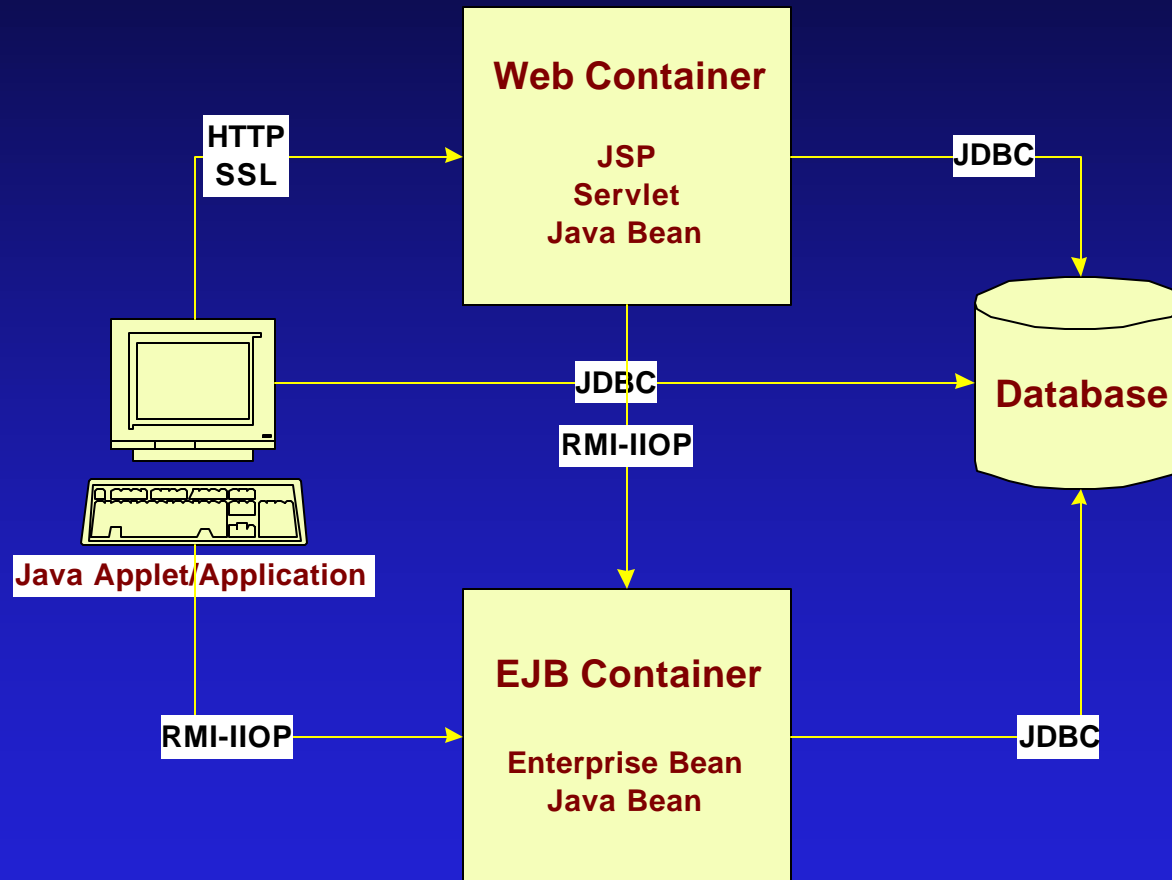
**Purpose:** To provide a persistent data store.

- } Relational Database
- } Object Database

# Dynamic HTML Alternatives



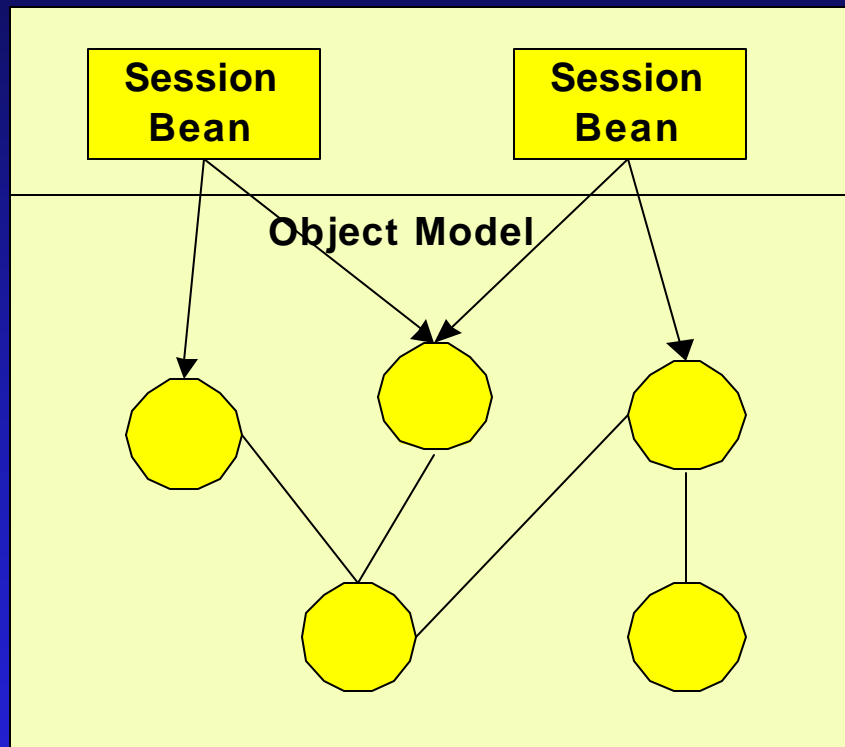
# Applet/Application Alternatives



# Advanced Architecture

- } DAO – Data Access Objects
- } Session Bean as Remote Interface
- } Facade – Facilitate Reuse of Business Logic
- } Object to Relational Mapping
- } Data Persistence

# J2EE Alternatives



# Facade

## (Presentation to Business Logic Mapping)

**Purpose:** To allow independence and reuse of the Presentation and Business Logic layers

- } **No Facade** – Direct access of Business Logic from Presentation Layer
- } **Single Point** – One Class access all Business Logic Services.
- } **Single Presentation, Multiple Proxy** – One Class for Presentation Logic accessing one proxy for each Business Logic component.

# Object to Relational Mapping

**Purpose:** To simplify the process of synchronising data between the relational database and the Object world.

- } **Entity Bean** – Container managed persistence, Bean managed persistence.
- } **Custom Code** – Hand written or generated
- } **Third Party Products**

# Data Persistence

**Purpose:** To provide a consistent view of data and potentially increase data access performance.

- } Entity Bean (JDO)
- } Database
- } Object to Relational Mapping Layer

# Trends

- } Performance - Stateless Session Beans
- } Reuseability - Business Logic in Standard Classes that are generated into EJBs or Corba Services.
- } Data Persistence – Generated Object to Relational Code or Enhanced Object Caching

# Summary

- } Remember Purpose of Each Container
- } Design and Code for Reuse
- } Build in Capability for Performance Testing

# Suggested Reading

- } [www.javasoft.com](http://www.javasoft.com)
- } Designing Enterprise Applications with Java 2 Platform, Enterprise Edition



# Questions