Designing Java EE Applications

Objectives

After completing this lesson, you should be able to do the following:

- Identify Java EE design patterns and their purpose
- Describe the Model-View-Controller (MVC) architecture
- Define the purpose and role of Struts
- Define the purpose and role of JavaServer Faces (JSF)

Realizing the Benefits of Java EE

To leverage the full benefits of Java EE, you must design applications that are:

- Portable: You should be able to redeploy the Java EE applications to different servers, databases, and so on.
- Scalable: Web applications should be able to handle large numbers of users.
- Maintainable: A minimum amount of coding should be necessary for a new business rule.
- Reusable: For example, a class that processes credit cards should be reused by multiple applications.
- Simple: The business need should be solved with the least amount of complexity.

Java EE Issues

It is important to follow certain guidelines for the design and development of any new technology:

- Implement generally accepted design patterns and architectures.
- Focus on real business needs rather than simply adopting new technology.
- Employ the simplest technology to solve a business problem.

Design Patterns

Design patterns:

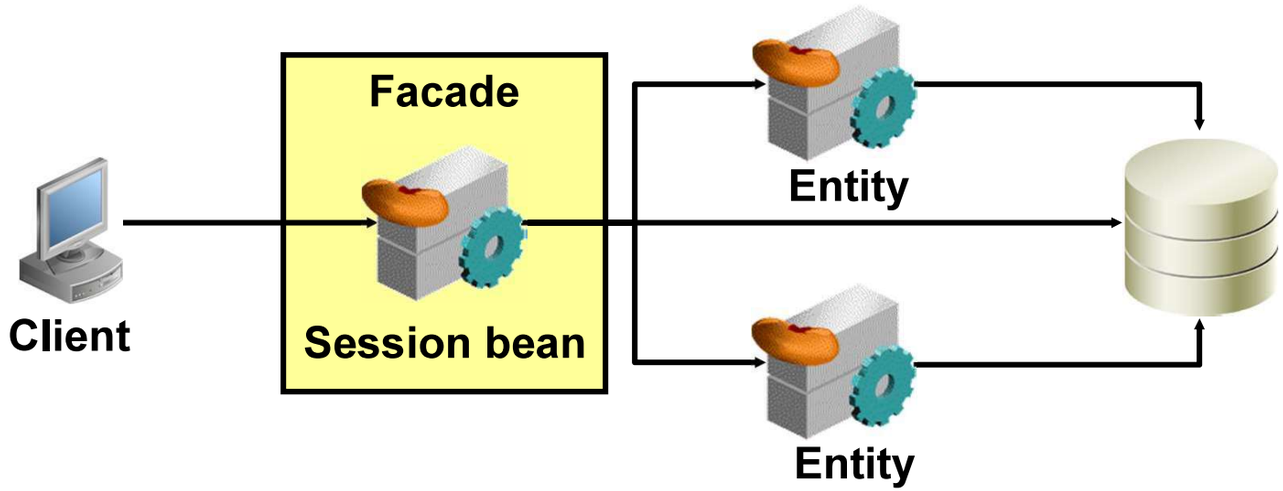
- Are a repeatable process of design addressing a specific issue or problem in the development of a system
- Are predictable in their function and are consistent in their pros and cons
- Are concepts that allows for a common method to quickly and effectively communicate complex ideology to others
- Allow for the accelerated education of the concept for those new to it

Java EE Design Patterns

The generally accepted design patterns include (but are not limited to) the following:

- Presentation-tier patterns:
  - Intercepting filter
  - Controller servlet (as used in MVC)
- Business-tier patterns:
  - Service Locator
  - Data Transfer Object
  - Session Facade

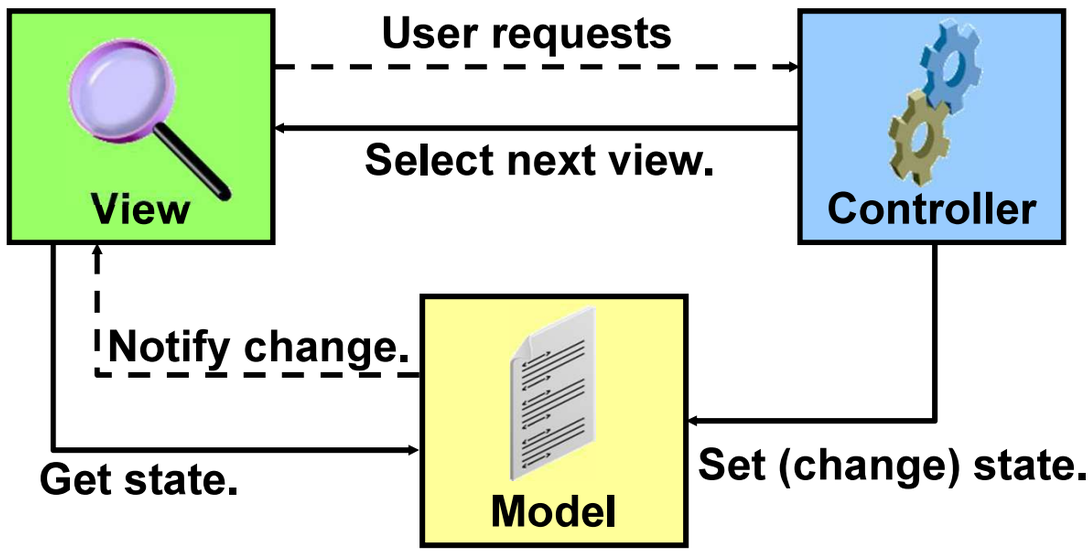
Implementing a Session Facade Pattern



A Session Facade pattern:

- Provides clients with a single application interface
- Contains centralized, complex interactions between lower-level business components
- Decouples components to yield a flexible design

Generic MVC architectural design pattern



The Model

- The model represents the enterprise data and business rules that handle access and updates.
- You can simplify the model by using two mechanisms called “facade class” and “command pattern.”
  - A facade encapsulates, hides the complexity of, and coordinates the operations between cooperating classes.
  - A command pattern encapsulates each application function in a separate class.
- The model is often implemented using Java beans and EJBs.

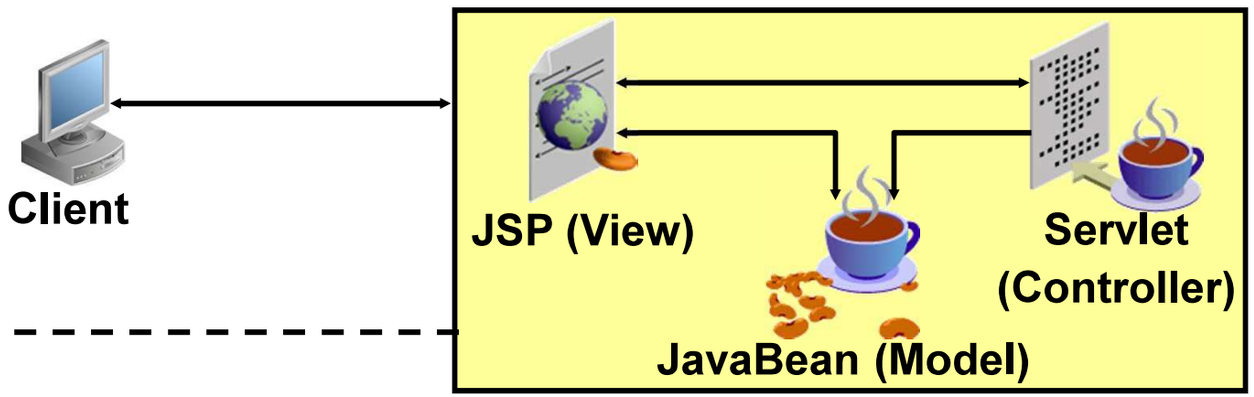
The View

- The view focuses on presentation and is responsible for maintaining consistency between data presentation and model changes. It enables:
  - Presentation to be changed without altering programming logic
  - Development by Web page authors having only visual design skills
- The view is commonly implemented using JSPs.

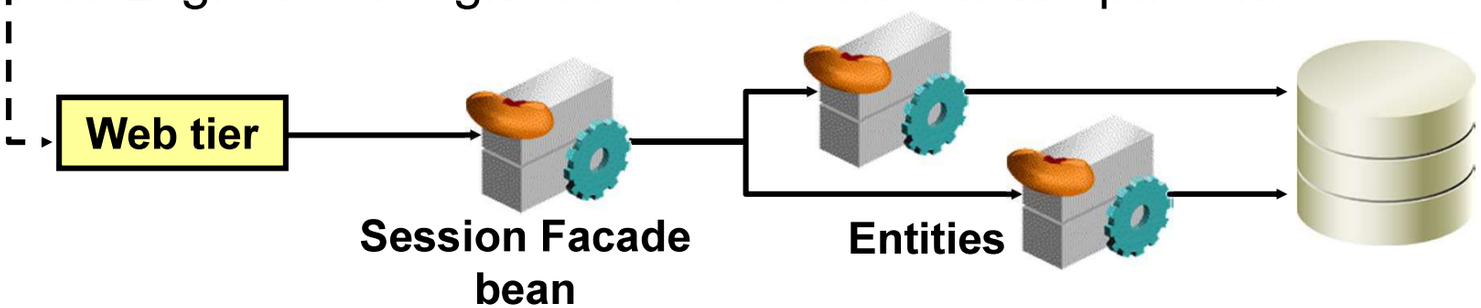
The Controller

- The controller provides interaction with the client, serving as a "glue" between the model and the view.
- The controller:
  - Interprets user requests and controls business objects to fulfill these requests
  - Removes navigation coding from the view
  - Can be implemented in the client, Web, or EJB tier, or in a combination of these tiers
- The controller is usually implemented as a servlet.
- Struts and JSF are two widely used implementations.

- Web-tier design: Use an MVC design pattern.



- Business Logic tier design: Use a Session Facade pattern.



Quiz

If you implement the MVC design pattern, the controller:

1. Adds navigation and implements the view
2. Can be implemented in the client, Web, or EJB tier, or in a combination of these tiers
3. Handles the input event from the user interface
4. Processes and responds to events, typically user actions, and may invoke changes on the model

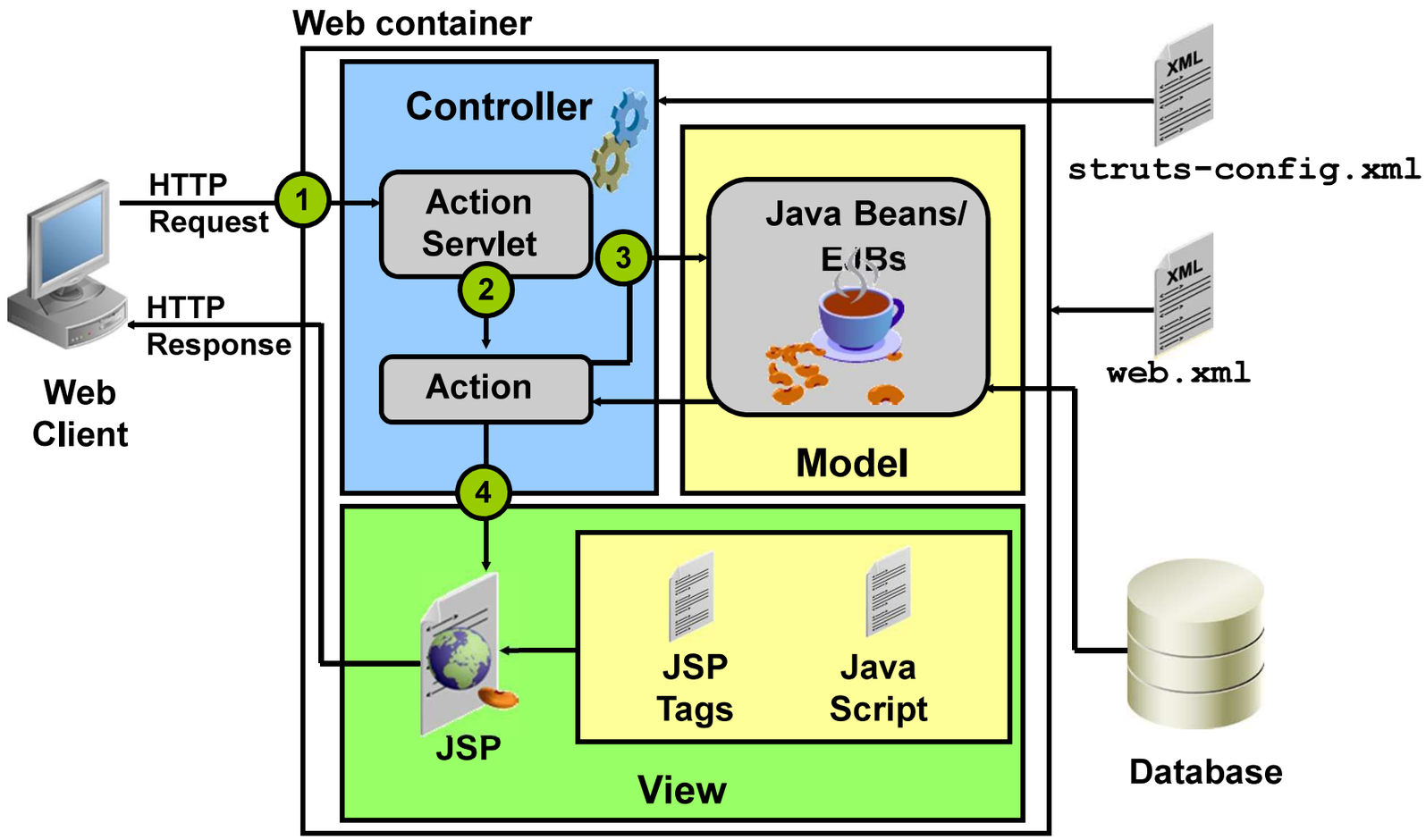
Struts: Overview

- Struts is a framework for implementing the MVC design pattern in Java EE applications.
- Struts:
  - Is an open-source project from the Apache Software Foundation
  - Is implemented by an XML-driven controller servlet
  - Uses standard Java EE Web technologies—servlets, JSPs, and tag libraries

Struts Components

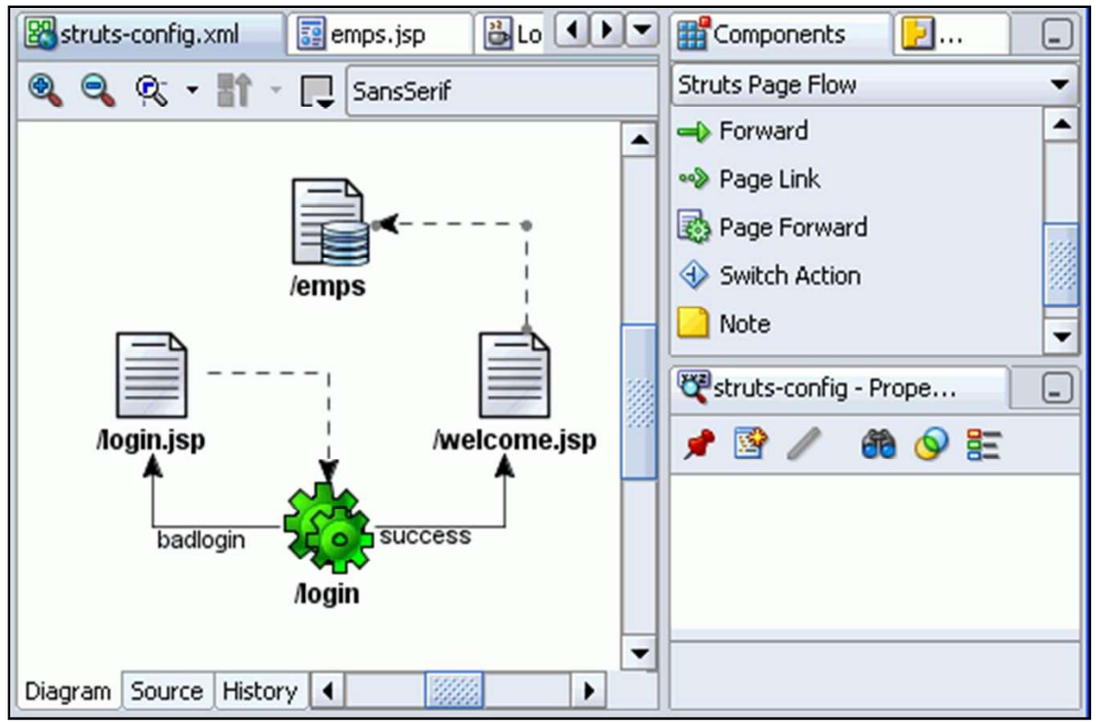
- *Action Servlet*: Receives user inputs and state changes and issues view selections
- *Action*: Interacts with the model to execute a state change or query
- *Form Beans*: Passes data (from forms) into JavaBeans for reuse and validation
- *Action Forward*: Stores the path to a page where the user is sent
- *Action Mapping*: Tells the servlet how each request URI is mapped to an Action
- `Struts-config.xml`: Contains the definitions of the Struts components used in an application

Struts Architecture



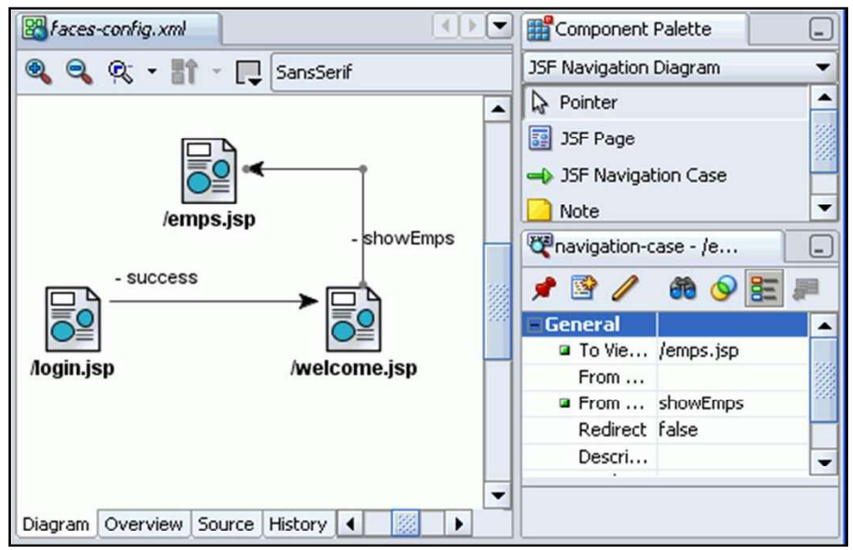
Struts Page Flow Design

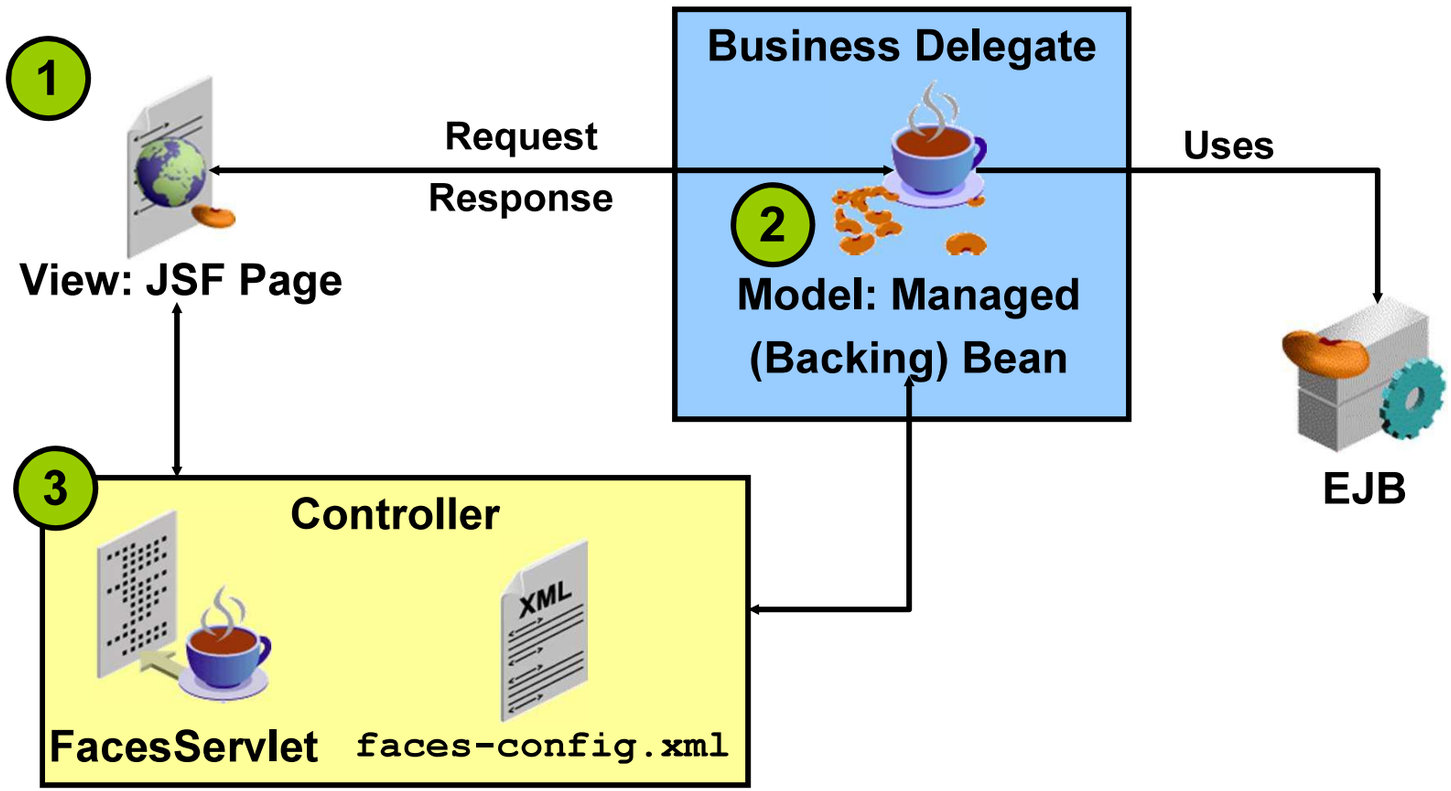
Use the Struts Page Flow diagram to create the top-level page flow design of a Struts application:



JSF provides the same benefits as the Struts architecture, including:

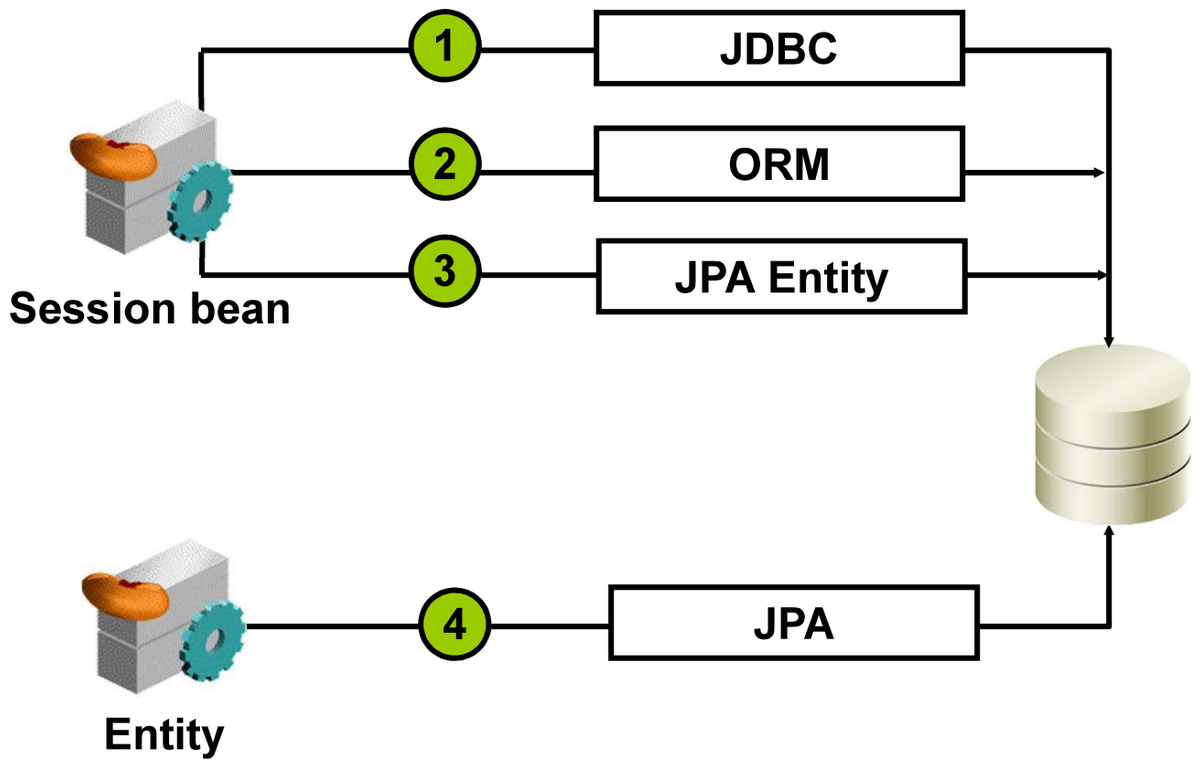
- Controller servlet (MVC implementation)
- Declarative and visual design in JDeveloper
- XML configuration file (`faces-config.xml`)




1
Request
Response
View: JSF Page
Business Delegate
2
Model: Managed
(Backing) Bean
Uses
EJB
3
Controller
XML
FacesServlet
faces-config.xml

Selecting a Persistence Strategy

Persistence techniques for Java EE applications:



Quiz

Struts is a framework for implementing the Session Facade design pattern in Java EE applications:

1. True
2. False

Summary

In this lesson, you should have learned how to:

- Identify Java EE design patterns and their purpose
- Describe the Model-View-Controller (MVC) architecture
- Define the purpose and role of Struts
- Define the purpose and role of JavaServer Faces (JSF)