COS30041 Creating Secure and Scalable Software

Lecture 05a Web User Interface

Learning Objectives

- After studying the lecture material, you will be able to
  - Understand and describe ideas of building a presentation layer for the Web front end
  - Discuss the advantages of having a presentation layer

Pre-requisites

- Some concepts in Object Oriented Programming
- Some design concerns in OOP [coupling, cohesion, single-minded components]
- Some web experiences
- BLL objects
- DAL objects
- MVC pattern

Outline

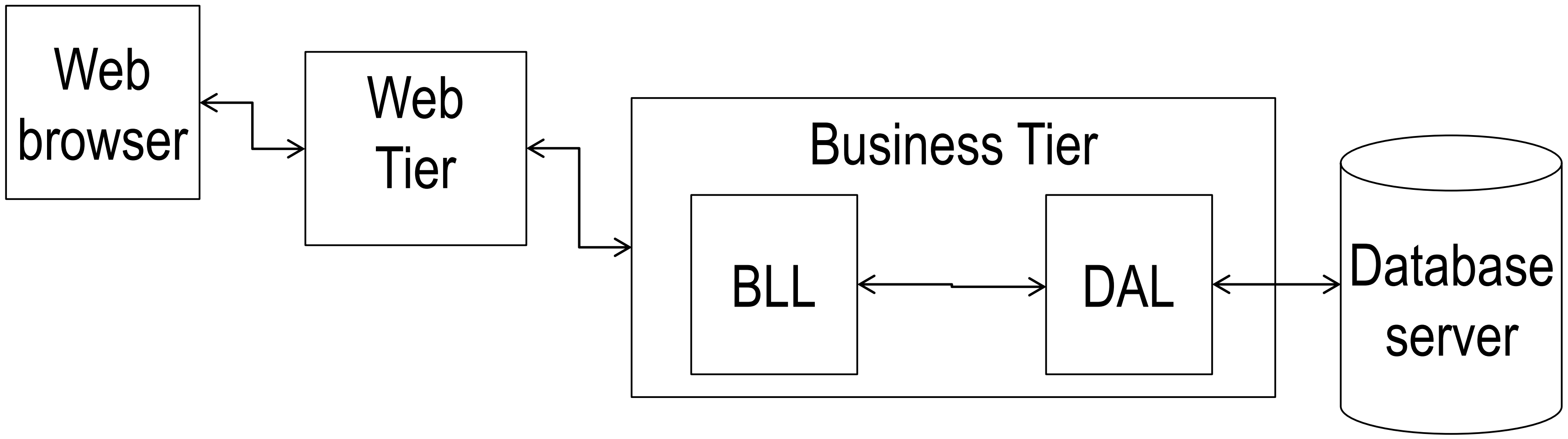
- A simple case study
- Another case study
- Something to think about?

A Simple Case Study – Logon page

- A simple log on action involves the following
  - logon page prompts for username and password pair
  - retryLogon page when username and password do not match
  - mainmenu page when logon is successful
  - logout page when user wants to logout
  - [**more advanced feature**] After three unsuccessful attempts, the user is blocked from future access unless they reset their password

Things to consider

- Where do we store username and password pair?
  - ☐ Permanently or Temporarily
- Where do we authenticate the user credentials?
- Where do we control the flow of the “web pages”?



BLL: Business Logic Layer
DAL: Data Access Layer

Enterprise Architecture (Recap from Lec 1)

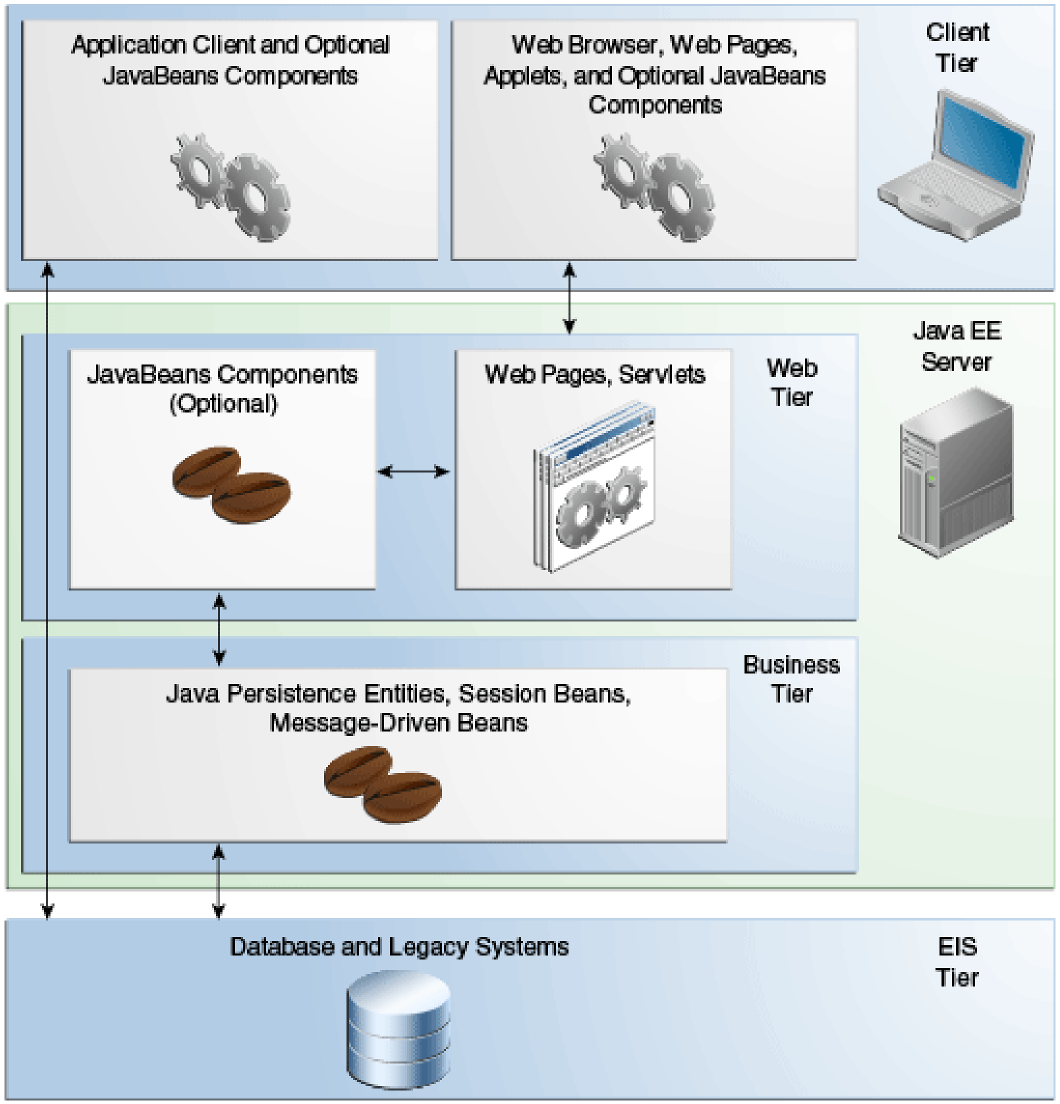

Application Client and Optional JavaBeans Components
Web Browser, Web Pages, Applets, and Optional JavaBeans Components
Client Tier
JavaBeans Components (Optional)
Web Pages, Servlets
Web Tier
Java EE Server
Java Persistence Entities, Session Beans, Message-Driven Beans
Business Tier
Database and Legacy Systems
EIS Tier

Fig.1-4 from [JEE7T]

Technology Issues

- Persistence – Database Table Schema
- DAL – Entity classes / objects
- BLL – Business objects [Session Beans; EJB]
  - Stateless / Stateful / Singleton
- Using the MVC approach
  - Model (?) -- View (?) -- Controller (?)
  - Data Access Layer -- Presentation Layer -- Business Layer

Technology Issues (cont'd)

- Web Server
  - How to get inputs from the web pages?
    - Where do we store these information?
  - How to determine whether the login is successful or not?
    - How to communicate to the BLL objects and obtain results from them?
  - How to control the flow of the pages based on the BLL decision and user selection?
  - How to display information related to user on "web pages"?
- Using the MVC approach
  - Model (?) -- View (?) -- Controller (?)

What is required?

Web UI

- A collection of “web pages” – displaying dynamic info
- A way to communicate user input to web server
- A way to display user info in web pages

Web Server

- A way to communicate to BLL objects
  - Send requests
  - Get responses
- A scheme to control the flow of the “web pages” based on
  - Authentication results
  - User selection

Another Case Study – Displaying Items

Example 1: Display a list of bank accounts (only account id) owned by a particular customer?

- Considerations
  - Do we use embedded SQL (“`SELECT AccountId FROM AccountDB WHERE UserId = ?`” `+` ...) in Web server to communicate to the Database server directly to get the required list and display?
  - Do we get a list of EVERY accountId in the bank (from the BLL object) for the Web server to filter and then display?
  - Do we filter out the list (in the BLL object first) and just return the required accountIds to the Web server for display?

Another Case Study – Example 1 (cont'd)

- Some possibilities
  - A string with pre-defined delimiter to hold all selected account ids?
    e.g. "`accountId1, accountId2, …`"
  - A list of selected account ids
    e.g. "`ArrayList<String> accountIds`"
  - A list of selected accounts
    e.g. "`ArrayList<AccountDTO> accounts`"

Another Case Study – Example 2

Example 2: Display a list of bank account details (e.g. account id with balance) owned by a particular customer?

- Considerations [same as before]
  - Embedded SQL ?
  - Web server to filter ?
  - BLL objects to filter ?

Another Case Study – Example 2 (cont'd)

- Some possibilities [similar to example 1]
  - ☐ A string with pre-defined delimiters to hold the info (e.g. "`accountId1, balance1; accountId2, …`")
  - ☐ A list of selected account ids and a list of selected balance (e.g. "`ArrayList<String> accountIds; ArrayList<Double> balance;`")
  - ☐ A list of selected accounts (e.g. "`ArrayList<AccountDTO> accounts`")
  - ☐ A list of selected details (account id and balance only) (e.g. "`ArrayList<DetailDTO> details`" where Detail holds only accountId and balance)

Technology Issues

- Persistence, DAL and BLL – the same as before
- Web Server
  - Same as before
  - Where do we store all such items on the web server?
  - How to display a list of items on “web pages”?
  - How to display individual instances of each item on “web pages”?

What is required?

Web UI

- A way to access a list of items stored in web server
- A way to display individual items from the list

Web Server

- A way to store a list of items from BLL objects
- A way to communicate the list with Web UI components

Web UI – Further Issues

- Validate user inputs on the spot
- Display error message if any
- Adding colour support for selected messages

References

- Java EE 7 Tutorial, Chapters 7 and 10