ASDV 2620, Web App Dev II
Midterm Exam Study Guide

1. Under Modules study and practice extensively the posted PDFs:

1.pdf

2.pdf

3ManagedBeans_1.pdf

4_bussinesLogic.pdf

5_managedBeans_3.pdf

6_managedBeans_3.pdf

7_navigation1.pdf

8_navigation2.pdf

9_navigation3.odt

10-EL-1.pdf

11-EL-2.pdf

12-EL-4.pdf

13-EL-5.pdf

16-Converters-1.pdf

2. **Horstmann textbook: Study Chapters 1, 2, 3, 4**

3.

1. What is a CDI Bean in JSF Framework? A CDI bean is a POJO, plain old java object, that has been automatically instantiated by the CDI container, and is injected into all, and any qualifying injection points in the application.

2. What is the annotation used for the declaration of a CDI bean ? `@Dependent`

```
public class AnotherPojoImp implements MyPojo{

    @Override
    public String getMessage() {
        return "Hello CDI 2.0 from AnotherPojoImp";
    }
}
```

3. What is the Bean Scope Application , Request , Session,None, Dependent ?

The application scope creates the bean instance for the lifecycle of a ServletContext, and the websocket

scope creates it for a particular WebSocket session.

The request scope creates a bean instance for a single HTTP request

The session scope creates a bean instance for an HTTP Session

The default scope if none is specified; it means that an object exists to serve exactly one client (bean) and has the same lifecycle as that client (bean).

Dependent means that an object exists to serve exactly one client (bean) and has the same lifecycle as that client (bean).

4. Properties messages often , have variable parts that need to be filled. How can we do this? i.e. **currentScore=Your current score is {0}.** (Look at quiz lab)

5. If you make a String property of a CDI Bean public do you need a getter to display in a JSF page or you do not need a getter? Test this. No.

6. What does the <h:commandButton action=”?????” /> attribute action do? Can either be used for navigation by replacing “?????” with the name of a xhtml or can be used to call methods from a bean. Expression language.

7. Implementing Business Logic in a Web Application we create java interfaces in Business Logic and then we declare placeholders-variables of the name of the interface inside CDI beans. Why we do this? Using interfaces and placeholder values with CDI beans is good practice and allows your programs to achieve modularity, maintainability, and flexibility. Also makes your code more robust and testable.

8. Can we inject Beans into beans? Look at the inject annotation to inject EJB Beans into CDI Beans. Yes you can.

9. Be able to change Locales in JSF Framework. (Look at quiz lab)

10. The JSF Life Cycle defines six distinct phases and one of them is Process Validations Know what each cycle does.

Phase 1: Restore view

JSF begins the restore view phase as soon as a link or a button is clicked and JSF receives a request.
During this phase, JSF builds the view, wires event handlers and validators to UI components and saves the view in the FacesContext instance. The FacesContext instance will now contain all the information required to process a request.

Phase 2: Apply request values

After the component tree is created/restored, each component in the component tree uses the decode method to extract its new value from the request parameters. Component stores this value. If the conversion fails, an error message is generated and queued on FacesContext. This message will be

displayed during the render response phase, along with any validation errors.
If any decode methods event listeners called renderResponse on the current FacesContext instance, the JSF moves to the render response phase.

Phase 3: Process validation

During this phase, JSF processes all validators registered on the component tree. It examines the component attribute rules for the validation and compares these rules to the local value stored for the component.
If the local value is invalid, JSF adds an error message to the FacesContext instance, and the life cycle advances to the render response phase and displays the same page again with the error message.

Phase 4: Update model values

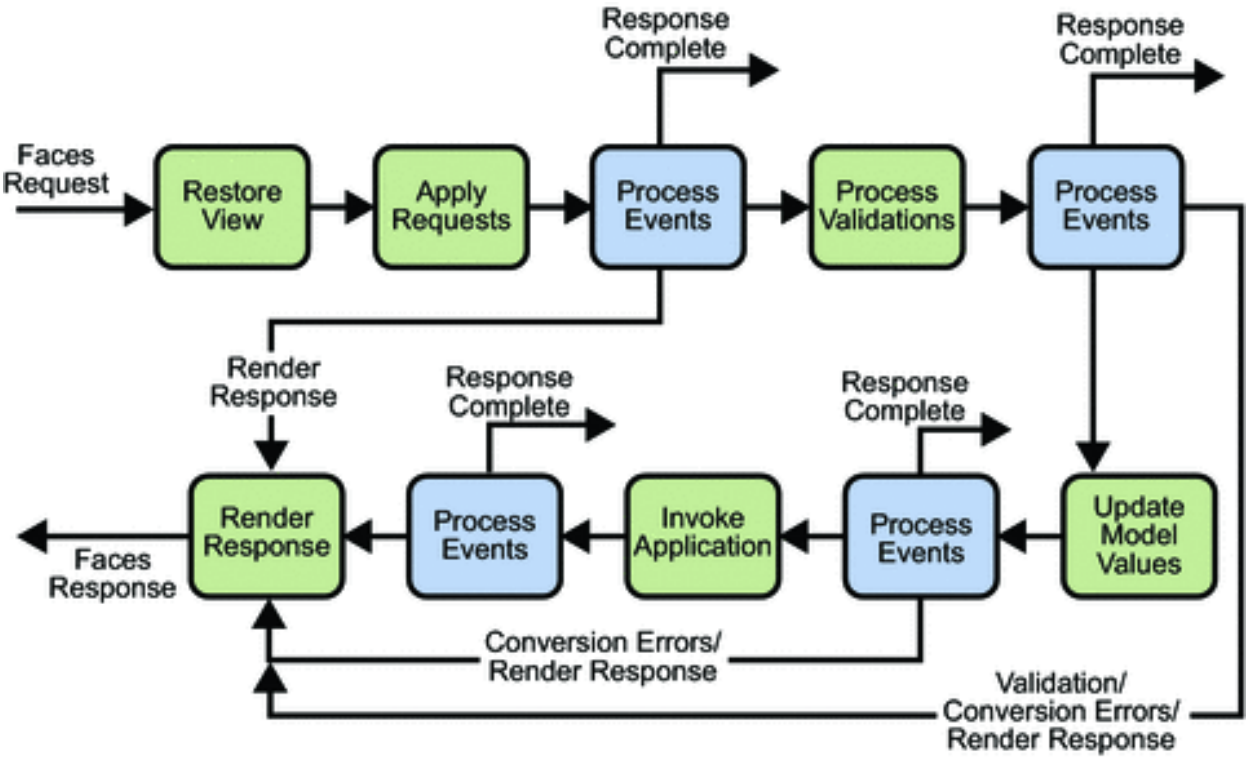
After the JSF checks that the data is valid, it walks over the component tree and sets the corresponding server-side object properties to the components' local values. JSF will update the bean properties corresponding to the input component's value attribute.
If any updateModels methods called renderResponse on the current FacesContext instance, JSF moves to the render response phase.

Phase 5: Invoke application

During this phase, JSF handles any application-level events, such as submitting a form/linking to another page.

Phase 6: Render response

During this phase, JSF asks container/application server to render the page if the application is using JSP pages. For initial request, the components represented on the page will be added to the component tree as JSP container executes the page. If this is not an initial request, the component tree is already built so components need not be added again. In either case, the components will render themselves as the JSP container/Application server traverses the tags in the page.
After the content of the view is rendered, the response state is saved so that subsequent requests can access it and it is available to the restore view phase.



3. **JSF Architecture Questions**

11. The Java EE / Jakarta platform has a primary goal to simplify the development of enterprise-class applications through an application mode,l that is, Vendor Neutral and Component Based. What is the MVC architecture used for in Web Applications? (bean= component, C=jsf page, M=datababase logic). Efficiently relating the user interface to underlying data models and organizing to relate the application code. MVC is primarily used to separate an application into three main components: Model, View, and Controller.

12. What are the reasons to prefer Web Applications vs Desktop Applications or vice versa? (Look at 1.pdf, 2.pdf under modules) Desktop applications need to be downloaded while Web Applications can be opened through a browser and are hosted on a server. Desktop applications have the upper hand with 2d and 3d graphics, and also do not require an internet connection to be ran.

13. What are GET requests? A GET request, in simple terms, is a way for you to grab data from a data source with the help of the internet.

14. What are POST requests? POST is used to send data to a server to create/update a resource.

15. What is an EJB Container? EJB is a server-side software component that encapsulates business logic of an application. An EJB web container provides a runtime environment for web related software components, including computer security, Java servlet lifecycle management, transaction processing, and other web services.

16. What EL is used for? Can we call static methods using EL? Can we dynamically perform arithmetic, boolean and string operations? Etc. The primary function of EL in JSF is to connect the JSF view (usually XHTML markup) and the java-based back-end. You cannot invoke static methods directly in EL. Yes.

17. How Navigation works? Sample question:

Unanswered

Question 11

Assume the form below is inside index.xhtml.

Given the CDI bean below in class X, what is the action we can put in the command button?

Check all that apply.

```
<h:form>
    <h:commandButton  />
</h:form>
```

```
import javax.enterprise.context.RequestScoped;

@Named(value = "X")
@RequestScoped

public class X
{
private String[] resultPages =
{
"page1", "page2", "page3"
};
 public String x()
   {
    int index = (int) (Math.random() * 3);
    return resultPages[index];
 }
}
```

Correct Answer ☐ action="#{ X.x }"

☐ action="#{ x.x }"

Correct Answer ☐ action="#{X.x() }"

Compiling error in the form:

☐ <h:commandButton /> is empty of attributes.

18. The getter methods are called in which Phases? Apply Request Values and Render Response

19. Test Inheritance in beans. Put the code in.

Unanswered **Question 16** 0 / 4 pts

Assume the EL expression in some JSF page:

#{messageHandler.message}

which attempts to assess property message which is a property of SimpleController whose name is NOT messageHandler . What is possible to happen?

```
@RequestScoped
public class SimpleController2
extends SimpleController
{

}


@Named(value = "simpleController")
@RequestScoped
public class SimpleController
{

}
```

Correct Answer
- [ ] #{messageHandler.message} will crash the app if messageHandler is not the name of bean SimpleController and no such name is found declared in the faces-config.

Correct Answer
- [ ] #{messageHandler.message} will NOT crash the app if messageHandler is not the name of bean SimpleController and the name is found declared in the faces-config.

- [ ] the program will crash becuase you cannot extend beans.

Correct Answer
- [ ] #{messageHandler.message} will NOT crash if we add the line @Named(value = "messageHandler") to the bean SimpleController2.

20. EL sample question:

Unanswered **Question 17** 0 / 3 pts

Check all that apply.

#{X.y[0].z}

Correct Answer ☐ Access y on bean named X (which should return an array or list), then take first entry, then call getZ on that, then print it.

☐ Access y on bean named X (which should have a public array or list), then take first entry, then call getZ on that, then print it.

Correct Answer ☐ z could be a property of some other class embedded in bean named X.

Correct Answer ☐ The type of y is *instanceof* the class that contains z.

Think of **#{X.y[0].z}** as **#{employee.addresses[0].zip}**

employee is named X

addresses is y, an array whose we get the first element.

zip is z, that is embedded in Addresses

21. EL Sample question. Look at posted EL manual under Modules for more details.

Unanswered

Question 19 0 / 3 pts

Given the EL expression and the *purchases* bean check what is correct.

EL:

<h:outputText value="#{purchases.mediumItems[2] le purchases.mediumItems[1] ? 'yes' : 'no'}"/>

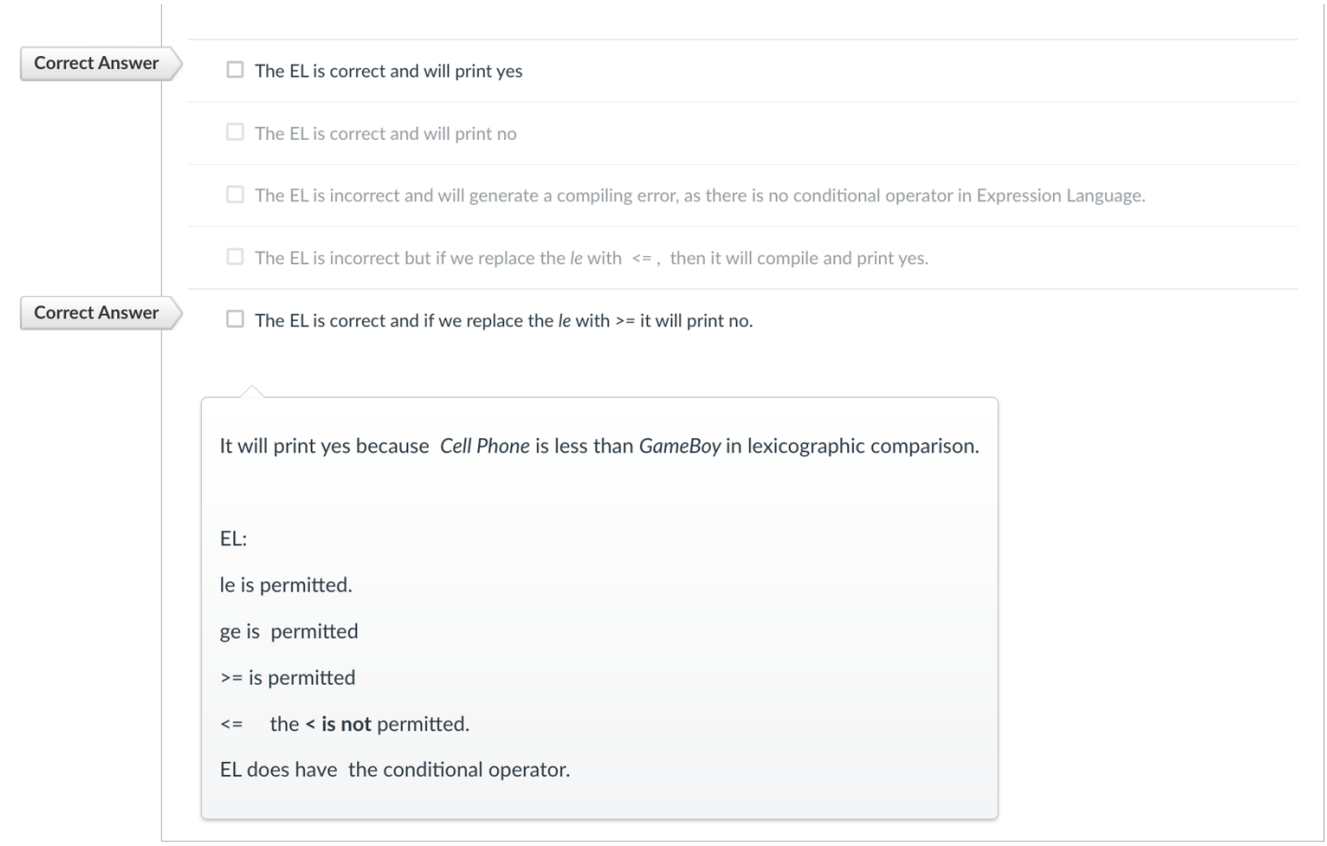
Bean:

```
@Named(value = "purchases")
@RequestScoped
public class Purchases
{
 private List<String> mediumItems= new ArrayList<>();

 public Purchases()
  {
  mediumItems.add("iPod");
  mediumItems.add("GameBoy");
  mediumItems.add("Cell Phone");
  }

 public List<String> getMediumItems(){ return mediumItems; }
}
```

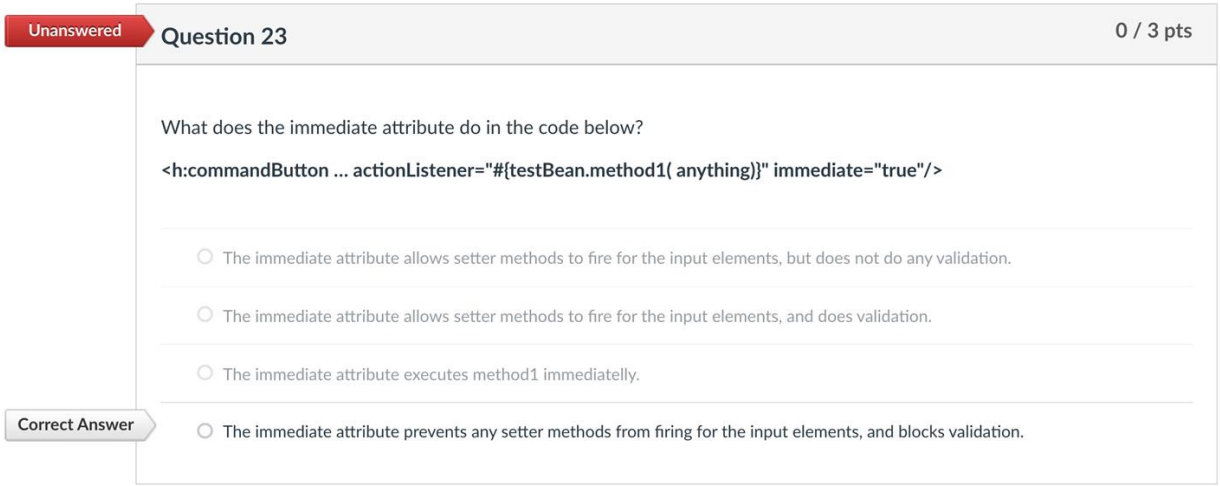
The outputText will be ‘yes’ because it uses the compareTo method of the Strings.



22. Navigation Rules and cases in config file:

Know how they work. What do you have to do to set them up in the config file and in the bean. (Look at quizBean lab)

23. What is the immediate attribute?



24. Converters. How do they work? They are essentially variable names to call back to. They are mainly used when using message properties files. Called by using {0}.

25. How custom validators work? What Interface we implement and how is it used? Custom validators are POJOs with the Validator interface implemented. They throw exceptions if the conditions of their validate method are not met.

```
@FacesValidator("statusValidate")
public class StatusValidate implements Validator
{

    @Override
    public void validate(FacesContext fc, UIComponent uic, Object t)
            throws ValidatorException
    {
        checkStatus(t);
    }

    public void checkStatus(Object value)
    {
        String s = value.toString();
        try
        {
            Integer.parseInt(s);
        }
        catch (Exception e)
```

Juneau Textbook

26. Study Chapter 3: The Basics of JavaServer Faces

27. The Labs we did in class with multi-components , select-one, select-many, radio buttons, check boxes, list boxes....Statistically and dynamically populated controls.

28. Building Sophisticated JSF Views with Components, page112

29. The Datatable, its usage and manipulation. Datatable is essentially a table that can autofill itself with values from a list. Kind of like a for each loop. Set a var for the table to call from (I.e supplier) then give it the value of a list you want it to call from (I.e #{supplierBean.getSuppliers()}. It will go through everything.

30. The facet tag. Facet is kind of like a header for a column in a table.

31. MP1, MP2, MP3, MP4