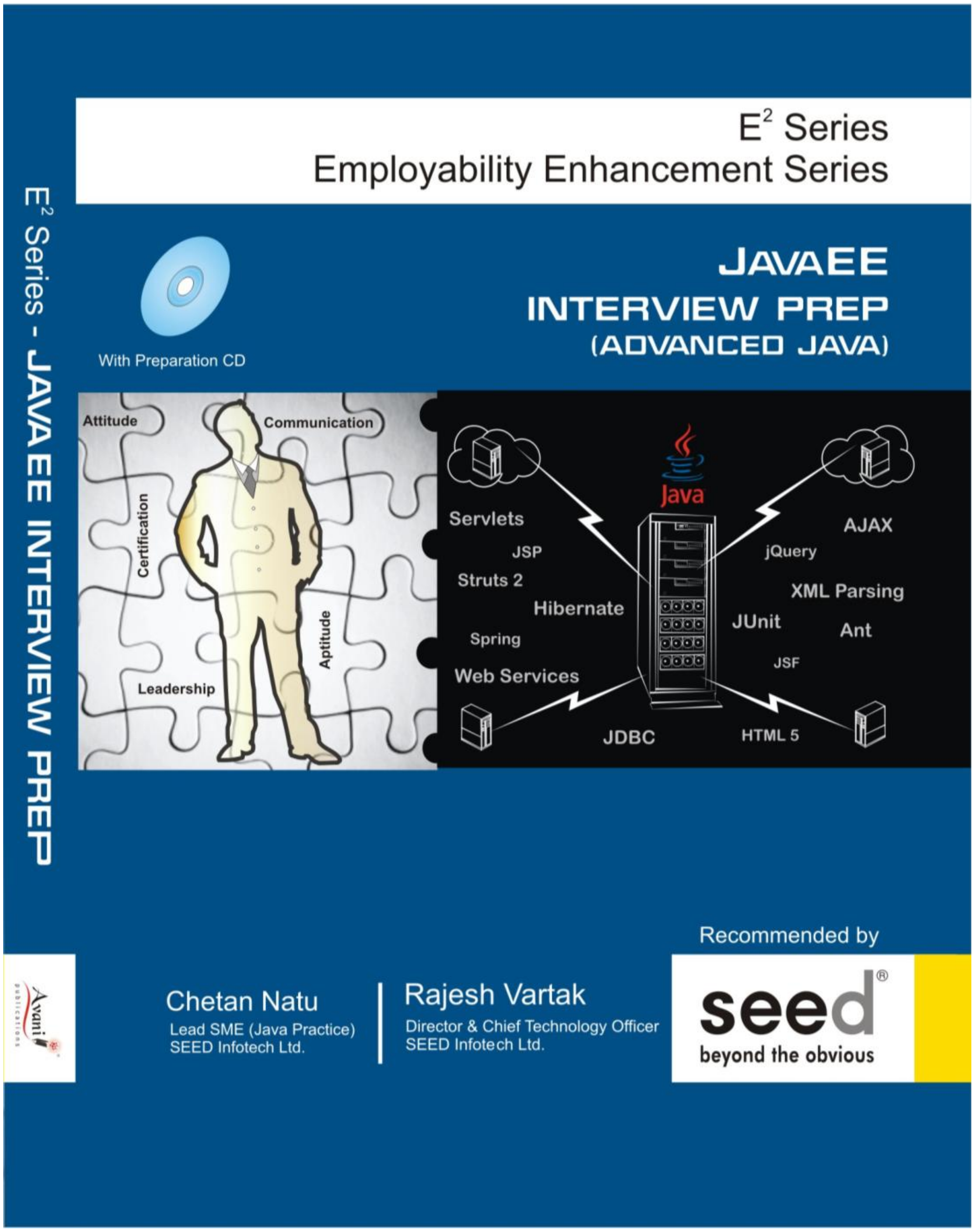


E^2 Series
Employability Enhancement Series

JAVAEE INTERVIEW PREP (ADVANCED JAVA)

With Preparation CD

E^2 Series - JAVAEE INTERVIEW PREP

Chetan Natu
Lead SME (Java Practice)
SEED Infotech Ltd.

Rajesh Vartak
Director & Chief Technology Officer
SEED Infotech Ltd.

Recommended by
seed®
beyond the obvious

Avani publications

1. What is a JDBC driver?

A JDBC driver is a Java program / Java API which allows the Java application to establish connection with the database and perform the database related operations.

There are different types of drivers created for different databases. For example: JDBC-ODBC bridge (available as part of standard JDK), Oracle thin driver etc.

2. How many different types of drivers are available in JDBC?

JDBC supports four types of drivers:

- JDBC-ODBC Bridge: Primarily used to connect with databases through ODBC mechanism, For example, MS-Access. Due to limitations of ODBC, it is recommended only for quick testing purposes.
- Native API: Created by database vendors, it uses native i.e. non-Java API to connect to the database. For example, Oracle OCI driver.
- JDBC-Net pure Java: Primarily used for enterprise applications to provide flexibility while using configurable Data Sources.
- Pure Java: Connects with database directly using API created in Java. Provides faster execution. For example, Oracle thin driver.

3. Tell me the name and type of JDBC drivers you have used in your project.

Based on the need of the application a suitable driver out of four types can be chosen.

Generally for quick testing purpose type-I (JDBC-ODBC Bridge) is used, while for production environment type-III (Java Net) is used, which in turn should use type-IV (Pure Java) drivers for actual database connectivity.

If you are doing a client-server based application, type II i.e. native driver also could be a good option.

Note: You need to know how JDBC driver configuration was done in your project.

Following table shows some useful information on this:

Driver Type	Configuration
JDBC-ODBC Bridge (Type I)	You will need to go into control panel on Windows machine and configure a DSN through Administrative Tools.
Native API (Type II)	You will need to install the client library file (`.dll`, `.jar`) file on the client machine to enable the client to invoke calls on the database.
Java Net (Type III)	No explicit configuration required on the client side, but a Data Source and optionally a connection pool will have to be configured on a central Server (typically an application server).
Pure Java (Type IV)	Vendor supplied library (`.jar`) file will have to be made available on the client side (through `classpath`) to enable database calls.

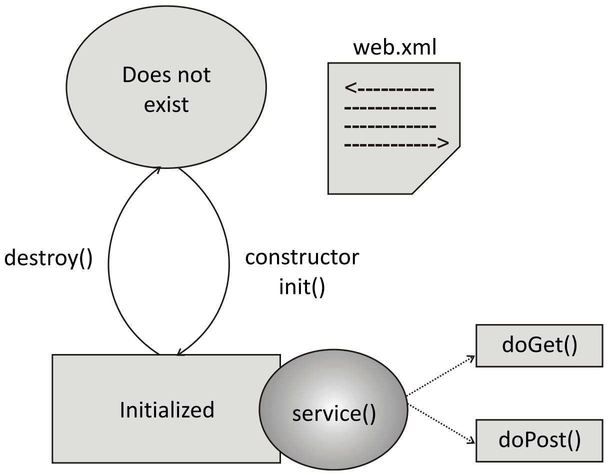
4. **Which is the fastest JDBC driver?**

Type 4 is the fastest driver, as it is written in such a way that it does not need any intermediate translation into database specific instruction set. It directly opens a socket connection with the database and establishes communications without any intermediate dependencies that are prevalent in other driver types.

5. **How do I load a JDBC driver?**

With Java versions till 5.x, a developer has to explicitly load a driver using the static method `Class.forName (<fully.qualified.name.of.driver>)` of class `Class`. With jdk 5.0 and above this not explicitly required. Depending on the URL provided in the `getConnection()` method of `DriverManager`, the class loader loads the appropriate class if it is available on the classpath.

1. Explain the Servlet LifeCycle.



Servlet goes through different methods in its life cycle as shown in the diagram.

Loading and Instantiation

The container loads the servlet during startup or when the first request is made for it. When a servlet will be loaded it is governed by `<load-on-startup>` value in `web.xml`. If `<load-on-startup>` value is positive, then the servlet loads at the startup along with the container. Otherwise it loads on the first request. Then container creates the instances of the servlet with `newInstance()` method.

Initialization and Readiness

`init()` is called by the servlet container before the servlet can service any request. The `init()` method is called only once throughout the life cycle of the servlet.If overridden, must call `super.init(config)` as its first statement.
Once initialized, the servlet is ready to serve any request using its `service()` method.

Destroying the Servlet

`destroy()` method is called when servlet is unloading to indicate that the servlet is being taken out of service. It frees the resources and does the cleanup tasks.

1. If JSPs are ultimately converted into servlets why create JSP?

Java servlets can create an HTML output to render a web page UI. The entire servlet code would be littered with `out.printIn` statements with embedded HTML syntax and Java code. It becomes very difficult to maintain such code.

JSP provides more clean way of separating rendering of view from navigation and flow control logic. As it looks more like HTML page, it would be easier for a web page designer to understand design the markup for rendering. It also helps in achieving separation of concerns as JSPs do not contain any Java code. Most popular MVC design pattern uses JSP as view and servlet as controller.

2. How can a JSP be created using XML syntax? Ⓐ

JSP syntax can also be written like a well-formed and valid XML document.

JSP syntax	XML syntax
`<%@ page attr list %>`	`<jsp:directive.page page_directive_attr_list />`
`<%@ includefile="relativeURL" %>`	`<jsp:directive.include file="relativeURL" />`
`<% code fragment %>`	`<jsp:scriptlet>code fragment</jsp:scriptlet>`

3. If JSP is converted into a servlet by the JSP Engine, where will I find the code written in JSP syntax elements within the servlets?

JSP Elements	Where it will be seen in generated servlet
Expression	Within the `service()` method.
Scriptlet	Within the `service()` method.
Comment	Comments are embedded in the same sequence in which they appear in the JSP.
Declaration	As part of the servlet class

1. What is `ActionMapping`?

Action mapping contains all the deployment information for a particular Action bean. This class determines where the results of the Action will be sent once its processing is complete.

In other words, it can be said that the `ActionMapping` encapsulates the `ActionMapping` tag specified in the `struts-config.xml` file. It is typically used by the `Action` class after the completion of the `execute()` method.

```
mapping.findForward(“controlString”);

// mapping is an ActionMapping object passed as parameter
to the execute() method.
```

The **`findForward(String)`** method directs the framework to look into the **`ActionMapping`** tag associated with the corresponding `Action` class and identify the URL associated with the “controlString” through the **`<Forward` …>** tag.

2. How is Action mapping specified?

We can specify the action mapping in the configuration file called `struts-config.xml`. Struts framework creates `ActionMapping` object from `<ActionMapping>` element of `struts-config.xml` file.

```
 <action path="/submit"
type="submit.SubmitAction"
        name="submitForm"
        input="/submit.jsp"
        scope="request"
        validate="true">
  <forward name="success" path="/success.jsp"/>
  <forward name="failure" path="/error.jsp"/>
 </action>
</action-mappings>
```

3. What is the role of `Action` class?

Contrary to popular belief, `Action` class is not a representation of a Model. The `Action` class performs the role of a gateway between the incoming HTTP request and the corresponding business logic to be executed as part of Model.

Enterprise applications may have business logic built inside components like Java Bean, an EJB, a native application or even a legacy system. `Action` class exposes the `execute()` method to link up view component of Struts with the business logic. In case of trivial (checking username and password) business logic, it is fine to write it as a part of `execute()` method of `Action` class.

```
public ActionForward execute(

ActionMapping mapping, ActionForm form,
             HttpServletRequest request,
HttpServletResponse response)
          throws Exception ;
```

`execute()` method performs the processing required to deal with this request, updates the server-side objects (Scope variables) that will be used to create the next page of the user interface and returns an appropriate `ActionForward` object.

4. What is the difference between page-centric and servlet-centric architecture?

Depending on the complexity and scope of the application, one can choose to an approach.

Page-centric Model	Servlet-centric Model
JSP acts as a controller.	Servlet acts as a controller.
No clear separation of responsibilities.	Clear separation of responsibilities.
Hard coded navigation flow. `<jsp:forward..>`	Configuration based navigation flows. `struts-config.xml`
Simpler and faster to create.	Complex to create.
Suitable for simple, small applications.	Suitable for complex enterprise applications.

1. **What is the difference between an Entity Bean and a Session Bean?**

   Session beans are used for implementing business process while Entity beans are used to manipulate database tables using Java objects.

2. **What is a message-driven bean?**

   A message-driven bean combines features of a session bean and a Java Message Service (JMS) message listener, allowing a business component to receive JMS. A message-driven bean enables asynchronous clients to access the business logic in the EJB tier.

3. **How does EJB Invocation happen?**

   The EJB invocation happens as follows:

   1. JNDI is used to retrieve the EJB Home reference from the Naming server using naming service.
   2. The EJB Home reference is held in a Home interface reference.
   3. A new EJB Object reference is created using the Home interface.
   4. An EJB Object is created (using `create()` method) on server.
   5. The EJB reference is held in a Remote interface reference.
   6. The Remote interface is used to invoke the business methods.

```
try
{
   Context initial = new InitialContext();
   Object objref =
   initial.lookup("SimpleConverter");
   ConverterHome home =
   (ConverterHome)PortableRemoteObject.narrow(objref,
    ConverterHome.class);
    Converter currencyConverter = home.create();
    BigDecimal param = new BigDecimal ("100.00");
    BigDecimal amount =
    currencyConverter.dollarToYen(param);
    System.out.println(amount);
    amount = currencyConverter.yenToEuro(param);
    System.out.println(amount);
    System.exit(0);       }
```

1. **What is JSF?**

   Java Server Faces (JSF) is a Java based server-side user interface component framework. It provides pre-built UI controls through tag libraries and an API for handling events, validations etc. JSF can generate graphics independent of devices. It can also be viewed as an MVC framework like Struts. JSF 2.0 provides very easy-to-use Ajax support. A better programming model and tag libraries make it easier for a developer to build and maintain web applications with minimal effort.

2. **What are the advantages of JSF?**

   The major benefits of JavaServer Faces technology are:

   - JavaServer Faces architecture makes it easy for experienced developer to build web applications faster.
   - The UI component lifecycle is managed entirely by the framework.
   - Has a clear cut separation of responsibilities between the presentation components and their underlying behavior.
   - Provides a rich architecture for managing component state, processing component data, validating user input, and handling events.
   - Works on a robust event handling mechanism similar to Swing.
   - High level of vendor independence through various rendering kits, pluggable components.

3. **What are disadvantages of JSF?**

   Though JSF has a better programming model and tag libraries, lot of work is done behind the scenes. Following are some of the disadvantages:

   - JSF is harder to understand and optimize
   - JSF approach towards MVC implementations seems to be more rigid.
   - JSF has a bigger learning curve.
   - JSF documentation is also not as good.

1. Have you faced problems while doing your project?

Possible intention

- To know if you have really worked on projects that you claim.
- To know the kind of exposure you have with respect to domain and type of customers.
- To know what kind of work you have really done so that interviewer can assess how you fit in the new role.

Insights

- Technical description of the problem [see the example answer uses many technical words; this indicates that you really know the details of the problem] and some more explanation for interviewer to understand is a must.
- Also explain what you did to resolve the issue. Again one more chance of marketing yourself.

Example answer

In our Java project, a .war file was not getting deployed on the weblogic application server. It was working fine in the development environment but when deployed in the QA environment it started behaving weird and exceptions were thrown. We tried our level best to debug and tried by changing some configuration settings like creating new domain, changing the database driver, making some changes to classpath.

Then we found that some libraries and old class files were interfering. Once we removed them, it started working smoothly. All of the team members including the lead, did quick meetings after that - thrice a day and resolved the issue on war footing.

Refer Q 27 for additional examples.

Anti pattern

- Do not say I never faced problems. It indicates you are lying or you do not have experience of real environment.

- Do not send a message like you were the only mighty soul who fixed all the issues that were encountered.

Interviewer's interpretation

- Real technical description and logical solution to the problem builds confidence of the interviewer that if one is acquainted with the problems already, he can be an asset to the organization.
- Your approach of getting solution creates impression of hard work, team play and passion.

EmployabilityEnhancementSeriesBooks

E^2 Series
www.eeseries.co.in

JAVAEE
INTERVIEW PREP
(ADVANCED JAVA)

This book is part of employability Enhancement Series (E2 Series). To get into software industry, the first step is clearing the interview. This book acts as a guide or companion for the preparation. It is based on the experience of training thousands of students every year and preparing them to perform their job effectively.

Employability = Graduation + Soft skills + Hard skills

This book covers most aspects of hard skills part for Software Development job in Java in a question and answer format. It includes various sections such as HTML, CSS, HTML5, JavaScript, jQuery, JDBC, Servlets, JSP, EJB, JMS, Struts framework, Hibernate framework, Spring framework.

Why this book is different?

- Backed up by experience of training job aspirants for more than decade
- More than 450 actual questions from interviews
- In-depth coverage of Java and related web application technologies
- Coverage of advanced topics such as design patterns, HTML5, jQuery, JSF, Web Services, Spring, Hibernate.
- Career building and employability enhancement strategies

Accompanying preparation CD contains

- Useful assorted articles on Java EE related application development
- Sample resumes for software developers
- Sample audio answers for questions on current affairs
- Preview of Resume to HR interview Prep book, Java interview Prep book
- Job hunting strategy
- How to strengthen your resume

Reader Speak

"This book truly works as a "vitamin dose" at the crunch time of interviews. It covers all the concepts in increasing level of complexity and allows for a quick refresher for basics". – **Smita, Project Manager, IT Projects**

"These books gave me most important thing - that is confidence". – **an aspiring Software Developer**