Chapter 1 : 3 - C H A P T E R -

Foundations of JSP Design Patterns and millions of other books are available for Amazon Kindle. Learn more Enter your mobile number or email address below and we'll send you a link to download the free Kindle App.

You can implement Helpers as regular Java classes. See Using Additional Classes or Beans for details. Composite Views A Composite View is a design pattern that creates an aggregate view from component views. Component views might include dynamic, modular portions of the page. This design pattern pertains to web application design when you are creating a view from numerous subviews. Complex web pages frequently consist of content derived from various resources. The layout of the page is managed independently of the content of its subviews. An included view is a subview that is one portion of a greater whole. Static content might consist of an HTML file. Dynamic content might be something like a JSP page. You can also include content at JSP translation time and runtime. View Creation Helpers Typically, web pages need to be reused and maintained over time. Use View Creation Helper beans when you need to display data as it is received. Examples of such information might be tables or sets of links. A View Creation Helper can be any bean or Java class. However, since they are specifically geared toward presentation in JSP pages, View Creation Helpers are typically tag handler classes. The View Creation Helper class provides a way to avoid placing Java code related to specific presentation features directly in the JSP file or Front Controller servlet. For instance, your web application might contain a catalog search that results in certain display results. View Creation Helper beans can be obtained and used in the same manner as any other beans from within a JSP file. Model Objects Model objects are Java objects that encapsulate application data inside a web application. For instance, in a shopcart e-commerce application, a model object might be an abstraction for a plush toy. This information is typically retrieved from database data is inefficient. Hence, you must design session data carefully. Typically, all requests from a single user go to a single server where the session information is stored. Hence, session information for a particular user must be shared among servers. This situation can result in slower performance if the web application must synchronize access to the session data. If the application is to be deployed on a load-balancing system, distributing session data is inefficient. Frameworks Frameworks are sets of design patterns, APIs, and runtime implementations intended to simplify the design and coding process for building new applications. Framework designers factor out common functions from existing applications and implement them using appropriate design patterns. A framework enables application developers to concentrate on the unique feature set required by their applications. Web application frameworks typically provide support for functions such as: The Struts package provides an integrated set of reusable components. The set includes a controller servlet, JSP custom tag libraries, and utility classes. The components are for creating user interfaces that can be applied to any web-based connection. Struts facilitates the standardization of workflow and the achievement of simplicity and reusability. For more on the Struts framework, see http: JATO unites familiar concepts with a often used design. Familiar concepts include display fields, application events, component hierarchies, and a page-centric development approach. For details on the JATO framework, see http: This set aims to simplify the building of Java web application graphical user interfaces GUIs. Furthermore, it enables tools and third-party component vendors to focus on a single component framework for JSP pages and servlets. Comprehensive support for internationalization and basic input validation is proposed. This support should ensure that developers include internationalization and input validation in their first releases. For more information on JavaServer Faces, see.

Chapter 2 : Free Book: EJB Design Patterns

This article, an excerpt from Foundations of JSP Design Patterns (Apress,), describes the View Helper pattern and shows how to build a few useful view helpers that you can add to your own toolkit.

The Model is to be represented by Javabean classes. This is often further dividable in Business Model which contains the actions behaviour and Data Model which contains the data information. Then, there are variations based on how actions and events are handled. The popular ones are: Request action based MVC: You have to gather, convert and validate the request parameters mostly yourself. But you end up with a simpler model and view wherein all the "raw" Servlet API is abstracted completely away. The Controller does this task and sets the gathered, converted and validated request parameters in the Model. All you need to do is to define action methods which works directly with the model properties. The state of the View for the subsequent requests is maintained in the session. This is particularly helpful for server-side conversion, validation and value change events. Learning an existing and well-developed framework takes in long term less time than developing and maintaining a robust framework yourself. Front Controller pattern Mediator pattern First, the Controller part should implement the Front Controller pattern which is a specialized kind of Mediator pattern. It should consist of only a single servlet which provides a centralized entry point of all requests. Simplest would be to use it as filename of the JSP. Map this servlet on a specific url-pattern in web. The parts register, login, etc are then available by request. Strategy pattern The Action should follow the Strategy pattern. The User itself is in turn a Data Model. The View is aware of the presence of the User. In this case, it should return an implementation of the Action interface based on the information provided by the request. For example, the method and pathinfo the pathinfo is the part after the context and servlet path in the request URL, excluding the query string. Let it for example return directly the request. Other patterns Those were the important patterns so far. To get a step further, you could use the Facade pattern to create a Context class which in turn wraps the request and response objects and offers several convenience methods delegating to the request and response objects and pass that as argument into the Action execute method instead. This adds an extra abstract layer to hide the raw Servlet API away. You should then basically end up with zero import javax. You can find a concrete example in this answer. This way you can evolve bit by bit towards a component based framework.

Chapter 3 : Decorator Design Pattern | Java Foundation

Find helpful customer reviews and review ratings for Foundations of JSP Design Patterns at www.nxgvision.com Read honest and unbiased product reviews from our users.

Java is a popular Object oriented programming language and has lots of design pattern and design principles, contributed by many developers and open source framework. As a Java programmer its expected from you to know OOPS concepts like Abstraction , Encapsulation, and Polymorphism , What is design pattern in Java, Some popular Java design pattern and most importantly when to use those design pattern in Java application. The purpose of asking design pattern interview question in Java is to check whether Java programmer is familiar with those essential design patterns or not. Design patterns in Java interviews are as important as multi-threading , collection, and programming questions. If you are senior or experienced Java programmer than expecting more complex and tough design pattern in Java interview e. Chain of Responsibility design pattern and solving real-time software design questions. Top Java design pattern questions and answers Here is my list of top 10 design pattern interview question in Java. I have also provided an answer to those Java design pattern question as a link. When to use Strategy Design Pattern in Java? Strategy pattern in quite useful for implementing set of related algorithms e. Strategy design pattern allows you to create Context classes, which uses Strategy implementation classes for applying business rules. This pattern follows open closed design principle and quite useful in Java. What is Observer design pattern in Java? When do you use Observer pattern in Java? This is one of the most common Java design pattern interview questions. Observer pattern is based upon notification, there are two kinds of object Subject and Observer. See What is Observer design pattern in Java with real life example for more details. Difference between Strategy and State design Pattern in Java? This is an interesting Java design pattern interview questions as both Strategy and State pattern has the same structure. If you look at UML class diagram for both patterns they look exactly same, but their intent is totally different. So Strategy pattern is a client driven pattern while Object can manage their state itself. What is decorator pattern in Java? Can you give an example of Decorator pattern? Decorator pattern is another popular Java design pattern question which is common because of its heavy usage in java. BufferedReader and BufferedWriter are a good example of decorator pattern in Java. See How to use Decorator pattern in Java for more details. When to use Composite design Pattern in Java? Have you used previously in your project? This design pattern question is asked on Java interview not just to check familiarity with the Composite pattern but also, whether a candidate has the real life experience or not. The Composite pattern is also a core Java design pattern, which allows you to treat both whole and part object to treat in a similar way. When the paint method is called on JPanel, it internally called paint method of individual components and let them draw themselves. On the second part of this design pattern interview question, be truthful, if you have used then say yes, otherwise say that you are familiar with the concept and used it by your own. By the way, always remember, giving an example from your project creates a better impression. What is Singleton pattern in Java? Singleton pattern in Java is a pattern which allows only one instance of Singleton class available in the whole application. Runtime is a good example of Singleton pattern in Java. Can you write thread-safe Singleton in Java? There are multiple ways to write thread-safe singleton in Java e. By the way using Java enum to create thread-safe singleton is the most simple way. See Why Enum singleton is better in Java for more details. When to use Template method design Pattern in Java? The Template pattern is another popular core Java design pattern interview question. I have seen it appear many times in real life project itself. Template pattern outlines an algorithm in form of template method and lets subclass implement individual steps. The key point to mention, while answering this question is that template method should be final, so that subclass can not override and change steps of the algorithm, but same time individual step should be abstract, so that child classes can implement them. What is Factory pattern in Java? What is the advantage of using a static factory method to create an object? Factory pattern in Java is a creation Java design pattern and favorite on many Java interviews. Factory pattern used to create an object by providing static factory methods. There are many advantages of providing factory methods e. See What is Factory pattern in Java and benefits for more details.

What is the difference between Decorator and Proxy pattern in Java? Another tricky Java design pattern question and trick here is that both Decorator and Proxy implements the interface of the object they decorate or encapsulate. As I said, many Java design pattern can have similar or exactly same structure but they differ in their intent. Decorator pattern is used to implement functionality on an already created object, while a Proxy pattern is used for controlling access to an object. You can also read Head First Analysis and Design to understand the difference between them. Use Setter injection to provide optional dependencies of an object, while use Constructor iInjection to provide a mandatory dependency of an object, without which it can not work. This question is related to Dependency Injection design pattern and mostly asked in the context of Spring framework, which is now become a standard for developing Java application. Since Spring provides IOC container, it also gives you a way to specify dependencies either by using setter methods or constructors. You can also take a look my previous post on the same topic. What is difference between Factory and Abstract Factory in Java I have already answered this question in detail with my article with the same title. The main difference is that Abstract Factory creates factory while Factory pattern creates objects. So both abstract the creation logic but one abstract is for factory and other for items. You can see here to answer this Java design pattern interview question. When to use Adapter pattern in Java? Have you used it before in your project? Use Adapter pattern when you need to make two class work with incompatible interfaces. Adapter pattern can also be used to encapsulate third party code so that your application only depends upon Adapter, which can adapt itself when third party code changes or you moved to a different third party library. By the way, this Java design pattern question can also be asked by providing an actual scenario. You can further read Head First Design Pattern to learn more about Adapter pattern and its real world usage. The book is updated for Java 8 as well so you will learn new, Java 8 way to implement these old design patterns. Can you write code to implement producer consumer design pattern in Java? The Producer-consumer design pattern is a concurrency design pattern in Java which can be implemented using multiple ways. If you are working in Java 5 then its better to use Concurrency util to implement producer-consumer pattern instead of plain old wait and notify in Java. Here is a good example of implementing producer consumer problem using BlockingQueue in Java. What is Open closed design principle in Java? Martin, popularly known as Uncle Bob in his most popular book, Clean Code. This principle advises that a code should be open for extension but closed for modification. At first, this may look conflicting but once you explore the power of polymorphism, you will start finding patterns which can provide stability and flexibility of this principle. One of the key examples of this is State and Strategy design pattern, where Context class is closed for modification and new functionality is provided by writing new code by implementing a new state of strategy. See this article to know more about Open closed principle. What is Builder design pattern in Java? When do you use Builder pattern? Builder pattern in Java is another creational design pattern in Java and often asked in Java interviews because of its specific use when you need to build an object which requires multiple properties some optional and some mandatory. See When to use Builder pattern in Java for more details Liskov Substitution Principle 4. Interface Segregation Principle 5. What is the difference between Abstraction and Encapsulation in Java? Even though both Abstraction and Encapsulation looks similar because both hide complexity and make the external interface simpler there is a subtle difference between them. Abstraction hides logical complexity while Encapsulation hides Physical Complexity. Btw, I have already covered answer of this Java interview question in my previous post as Difference between encapsulation and abstraction in Java , here are some more difference from that post. This was my list of 10 popular design pattern interview question in Java. Please let us know if you have any other interesting question on Java design pattern. Other Java interview questions post:

Chapter 4 : java - Example of Passive View design pattern using JSP and POJOs - Stack Overflow

Foundations of JSP Design Patterns gives you the tools to build scalable enterprise applications using JSP. While other books merely provide instruction on basic JSP and servlet development, this insightful guide goes a step further to offer a variety of best practices and design principles.

DOWNLOAD PDF FOUNDATIONS OF JSP DESIGN PATTERNS

Chapter 5 : This weeks giveaway : JSP Examples and Best Practices. (JSP forum at Coderanch)

Foundations of JSP Design Patterns gives you the tools to build scalable enterprise applications using JSP. While other books merely provide instruction on basic JSP and servlet development, this insightful guide goes a step further to offer a variety of best practices and design principles, enabling you to build your own scalable and extensible enterprise Java applications quickly and easily.

Chapter 6 : Top 18 Java Design Pattern Interview Questions Answers for Experienced | Java67

Each design pattern solves a certain problem associated with the view of application. Chapter Slays the foundations for these design patterns by describing the ModelÂ-.

Chapter 7 : Lynda â€“ Foundations of Programming: Design Patterns

** Andrew Patzer was the principal author of the best selling Professional Java Server Programming-among the first to cover J2EE technologies; JSP design patterns books should sell as well as the.*

Chapter 8 : java - Design Patterns web based applications - Stack Overflow

READ Foundations of Jsp Design Patterns by Andrew Patzer READ Foundations of Jsp Design Patterns Epub READ Foundations of Jsp Design Patterns Download v Slideshare uses cookies to improve functionality and performance, and to provide you with relevant advertising.