Java 2 EE Patterns (Business Delegate)

A developer is designing the presentation tier for a web application that relies on a complex session bean. The session bean is still being developed and the APIs for it are NOT finalized. Any changes to the session bean API directly impact the development of the presentation tier. Which design pattern provides a means to manage the uncertainty in the API?

Un desarrollador está diseñando la capa de presentación para una aplicación web que se basa en un complejo bean de sesión. El bean de sesión aún está en desarrollo y las API para ello no se han terminado. Cualquier cambio en el API del bean de sesión impactan directamente en el desarrollo de la capa de presentación. ¿Qué patrón de diseño proporciona un medio para gestionar la incertidumbre en la API?

- [ ] View Helper
- [ ] Front Controller
- [ ] Composite View
- [ ] Intercepting Filter
- [x] Business Delegate
- [ ] Chain of Responsibility

Squeaky Beans Inc. hired an outside consultant to develop their web application. To finish the job quickly, the consultant created several dozen JSP pages that directly communicate with the database. The Squeaky business team has since purchased a set of business objects to model their system, and the Squeaky developer charged with maintaining the web application must now refactor all the JSPs to work with the new system. Which pattern can the developer use to solve this problem?

Beans Squeaky Inc. contrató a un consultor externo para desarrollar su aplicación web. Para terminar el trabajo rápidamente, el consultor creó varias docenas de páginas JSP que se comunican directamente con la base de datos. El equipo encargado de Squeaky ha comprado ya un conjunto de business objects para modelar su sistema, y el desarrollador de Squeaky encargado de mantener la aplicación web debe ahora refactorizar todo el JSP para trabajar con el nuevo sistema. ¿Qué patrón se puede utilizar el programador para resolver este problema?

- [ ] Transfer Object
- [ ] Service Locator

- [ ] Intercepting Filter
- [x] Business Delegate

Java 2 EE Patterns (Service Locator)

A developer is designing a multi-tier web application and discovers a need to hide the details of establishing and maintaining remote communications from the client. In addition, the application needs to find, in a transparent manner, the heterogeneous business components used to service the client's requests. Which design patterns, working together, address these issues?

Un desarrollador está diseñando una aplicación web de varios niveles y descubre una necesidad de ocultar los detalles de establecimiento y el mantenimiento de las comunicaciones remotas desde el cliente. Además, la aplicación necesita encontrar, de manera transparente, los business components heterogéneos utilizados para atender las solicitudes de los clientes. ¿Qué patrones de diseño, trabajando juntos, abordar estas cuestiones?

- [ ] Business Delegate and Transfer Object
- [x] Business Delegate and Service Locator
- [ ] Front Controller and Business Delegate
- [ ] Intercepting Filter and Transfer Object
- [ ] Model-View-Controller and Intercepting Filter

The Squeaky Bean Company has decided to port their web application to a new J2EE 6 container. While reviewing the application, a developer realizes that in multiple places within the current application, nearly duplicate code exists that *finds enterprise beans.* Which pattern should be used to eliminate this duplicate code?

- [ ] Transfer Object
- [ ] Front Controller
- [x] Service Locator
- [ ] Intercepting Filter
- [ ] Business Delegate
- [ ] Model-View-Controller

Which two are characteristics of the Service Locator pattern? (Choose two.)

- [x] It encapsulates component lookup procedures.
- [ ] It increases source code duplication and decreases reuse.
- [x] It improves client performance by caching context and factory objects.
- [ ] It degrades network performance due to increased access to distributed lookup services.

Java 2 EE Patterns (Front Controller)

A developer is designing the presentation tier for a web application which requires a centralized request handling to complete common processing required by each request. Which design pattern provides a solution to this problem?

Un desarrollador está diseñando la capa de presentación para una aplicación web que requiere de un manejo centralizado de solicitud para completar el procesamiento común requerido por cada solicitud. ¿Qué patrón de diseño proporciona una solución a este problema?

- [ ] Remote Proxy
- [x] Front Controller
- [ ] Service Activator
- [ ] Intercepting Filter
- [ ] Business Delegate
- [ ] Data Access Object

A developer has created a web application that includes a servlet for each use case in the application. These servlets have become rather difficult to maintain because the request processing methods have become very large. There is also common processing code in many servlets because these use cases are very similar. Which two design patterns can be used together to refactor and simplify this web application? (Choose two.)

Un desarrollador ha creado una aplicación web que incluye un servlet para cada caso de uso de la aplicación. Estos servlets se han convertido en algo difícil de mantener debido a los métodos de procesamiento de solicitudes son muy extensos. También hay código de procesamiento común en muchos servlets, porque estos casos de uso son muy similares. ¿Qué dos patrones de diseño se puede utilizar en conjunto para refactorizar y simplificar esta aplicación web? (Elija dos opciones).

- [ ] Proxy

- [ ] View Helper
- [x] Front Controller
- [ ] Session Facade
- [ ] Business Delegate
- [x] Model-View-Controller

Which two are characteristics of the Front Controller pattern? (Choose two.)

- [ ] It simplifies remote interfaces to distributed objects.
- [x] It promotes cleaner application partitioning and encourages reuse.
- [x] It provides an initial point of contact for handling all related requests.
- [ ] It reduces maintainability due to the increased complexity of the design.
- [ ] It provides loosely coupled handlers that can be combined in various permutations.

Java 2 EE Patterns (Intercepting Filter)

A developer is designing a multi-tier web application and discovers a need to log each incoming client request. Which two patterns, taken independently, provide a solution for this problem? (Choose two.)

Un desarrollador está diseñando una aplicación web de varios niveles y descubre una necesidad de registrar cada petición de cliente entrante. ¿Qué dos modelos, tomados independientemente, dar solución a este problema? (Elija dos opciones).

- [ ] Transfer Object
- [ ] Service Locator
- [x] Front Controller
- [x] Intercepting Filter
- [ ] Business Delegate
- [ ] Model-View-Controller

A developer has created a special servlet that is responsible for generating XML content that is sent to a data warehousing subsystem. This subsystem uses HTTP to request these large data files, which are compressed by the servlet to save internal network bandwidth. The developer has received a request from management to create several more of these data warehousing servlets. The developer is about to copy and paste the compression code into

each new servlet. Which design pattern can consolidate this compression code to be used by all of the data warehousing servlets?

Un desarrollador ha creado un servlet especial que se encarga de generar el contenido XML que se envía a un subsistema de almacenamiento de datos. Este subsistema utiliza HTTP para solicitar estos grandes archivos de datos, que se comprimen por el servlet para ahorrar ancho de banda de la red interna. El desarrollador ha recibido una solicitud de gestión para crear varios más de estos servlets de almacenamiento de datos. El desarrollador trata de copiar y pegar el código de compresión para formar cada nuevo servlet. ¿Qué patrón de diseño puede consolidar este código de compresión para ser utilizado por todos los servlets de almacenamiento de datos?

- [ ] Facade
- [ ] View Helper
- [ ] Transfer Object
- [x] Intercepting Filter
- [ ] Composite Facade

Which two are characteristics of the Intercepting Filter pattern? (Choose two.)

- [ ] It provides centralized request handling for incoming requests.
- [ ] It forces resource authentication to be distributed across web components.
- [ ] It reduces coupling between presentation-tier clients and underlying business services.
- [x] It can be added and removed unobtrusively, without requiring changes to existing code.
- [x] It allows preprocessing and postprocessing on the incoming requests and outgoing responses.

Java 2 EE Patterns (Transfer Object)

A developer is designing a multi-tier web application and discovers a need to hide the details of establishing and maintaining remote communications from the client. In addition, because the business and resource tiers are distributed, the application needs to minimize the inter-tier network traffic related to servicing client requests. Which design patterns, working together, address these issues?

Un desarrollador está diseñando una aplicación web de varios niveles y descubre una necesidad de ocultar los detalles de establecimiento y el mantenimiento de las comunicaciones remotas desde el cliente. Además, debido a que los niveles de negocio y de

recursos se distribuyen, la aplicación necesita para minimizar el tráfico de red entre los niveles relacionados con el servicio de solicitudes de los clientes. ¿Qué patrones de diseño, trabajando juntos, abordar estas cuestiones?

- [ ] Front Controller and Transfer Object
- [ ] Front Controller and Service Locator
- [x] Business Delegate and Transfer Object
- [ ] Business Delegate and Intercepting Filter
- [ ] Model-View-Controller and Intercepting Filter

Which of the following apply to Transfer Object design pattern?

- [ ] It increases complexity by increasing the number of remote interfaces
- [x] It increases network performance by introducing one coarse grained remote call for multiple finer grained network calls
- [ ] It reduces network traffic by introducing one coarse grained remote call for multiple finer grained network calls
- [ ] It increase server throughput by utilizing the CPU better
- [x] It increases design overhead due to versioning issues

Java 2 EE Patterns (Model-View-Controller)

A developer is designing a web application that must support multiple interfaces, including:
an XML web service for B2B
HTML for web-based clients
WML for wireless customers
Which design pattern provides a solution for this problem?

- [ ] Session Facade
- [ ] Business Delegate
- [ ] Data Access Object
- [x] Model-View-Controller
- [ ] Chain of Responsibility

Which of the following application facilities can be controlled centrally by a Model View Controller (MVC) design pattern?
Select three choices.

- [ ] Database functionality
- [x] Security
- [x] Logging
- [x] Screen flow
- [ ] Accessing remote components