J2EE architektúra v praxi
Aplikačné modely

Peter Obert

9. 11. 2006

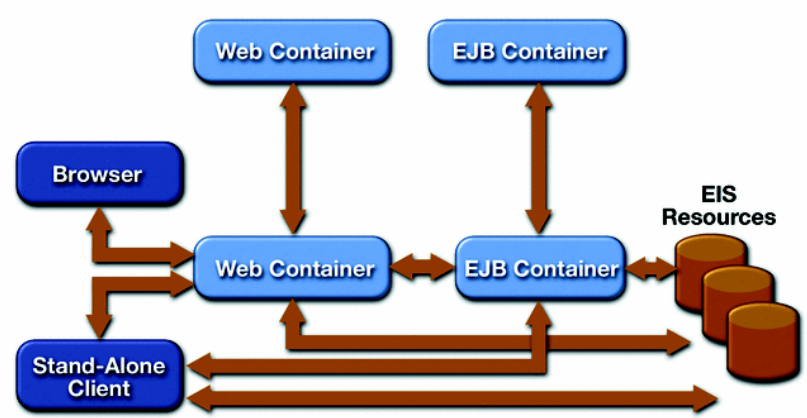
SOFTEC

Hlavné faktory vplývajúce na aplikačný design

- Potreba rýchlo a často meniť výzor aplikácie
- Rozdelenie aplikácie na prezentačné časti a časti aplikačnej logiky, tak aby bola zvýšená modularita aplikácie
- Zjednodušenie vytvárania jednotlivých častí, špecializácia vývojárov
- Potreba paralelného vývoja
- Nezávislosť jednotlivých častí od spôsobu a komponentov, ktorými sú vyvíjané
- Vytvorenie jednotného slovníka a pravidiel pre vytváranie aplikačnej logiky.
- Možnosť znovupoužitia a spájania komponentov aplikačnej logiky.
- Prevádzkovanie transakčných komponentov na rozličnom HW a softvérových platfomách nezávislých na prevádzkovanom SRBD.
- Schopnosť externalizovať vnútorné dáta bez potreby vedieť informácie o cieľovom spotrebiteľovi týchto dát tj. tzv vytváranie voľných väzieb

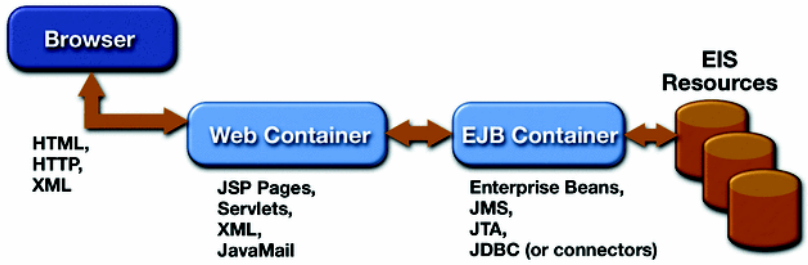
SOFTEC

Aplikačné scenáre v J2EE



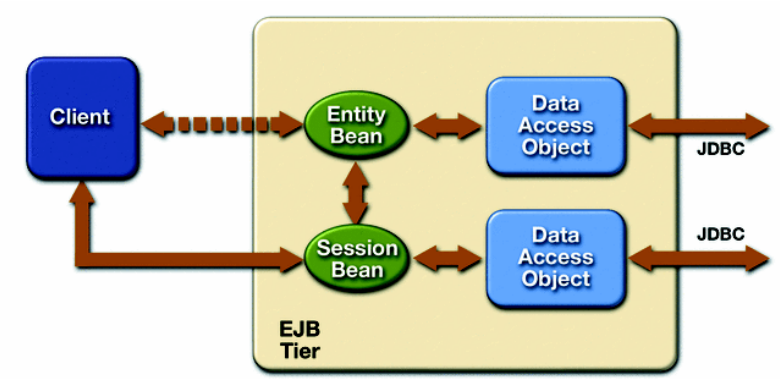
SOFTEC

Multivrstvová architektúra v J2EE



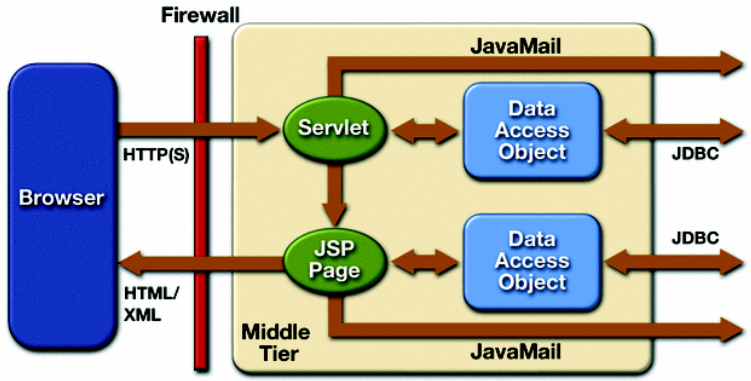
SOFTEC

Standalone a EJB centrický scenár



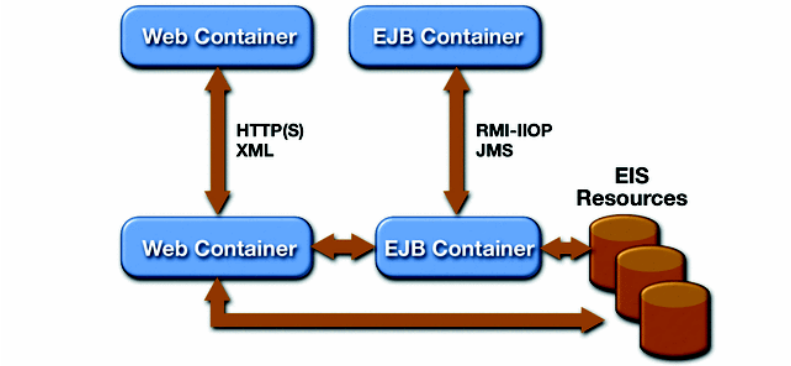
SOFTEC

Web centrický scenár



SOFTEC

B2B scenár



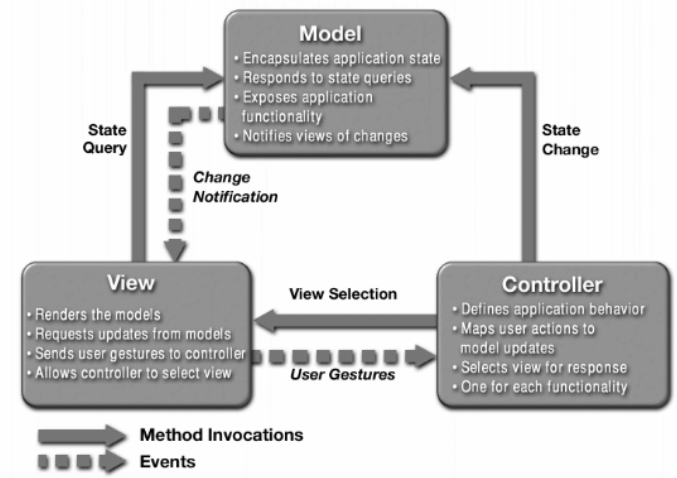
SOFTEC

MVC a reprezentácia v J2EE

- Model
  - reprezentuje aplikačné dáta a aplikačné pravidlá ako aj vykonáva samotné zmeny modelu.
    - nevizuálne JavaBeans, EJB, POJO
- View
  - pristupuje k aplikačným dátam cez model a riadi spôsob ich prezentácie
    - Html, JSP (Servlet), vizuálne JavaBeans... – dokument
- Controller
  - Transformuje interakcie do akcií ktoré sa budú vykonávať na modely a na základe zmeny stavu modelu určuje view zodpovendné za zobrazenie modelu.
    - Servlet (JSP)

SOFTEC

M-V-C



SOFTEC

MVC stratégie

- **MVC pre web klienta**
  - JSP stránky vytvárajú view, Servlet ako kontroler, Enterprise JavaBeans components as the model.(Java Pet Store).
- **Centralizovaný kontroler.**
  - Namiesto viacerých kontrolerov, spracovanie všetkých requestov riadi hlavný Servlet. The Front Controller pattern describes this strategy in more detail.
- **Decentralizovaný kontroler.**
  - View samotné rozhoduje o tom kto zobrazí model. Page controller pattern.
- **MVC pre Wireless klienta(mobilné t.f.).**
  - Smart Ticket aplikácia implementuje a predstavuje stratégiu.

SOFTEC

Ako implementovať MVC?

- Najčastejšie MVC implementácie, ktoré sa berú ako základ J2EE Web FE a Web aplikácií
  - Front Controller - Centralize application request processing
    - Struts
    - Java Server Faces
  - Page Controler distribute request processing
    - Tapestry, Velocity Templates
- (http://bdn.borland.com/article/borcon/files/6000/paper/6000.html)
- (https://equinox.dev.java.net/framework-comparison/WebFrameworks.pdf)

SOFTEC

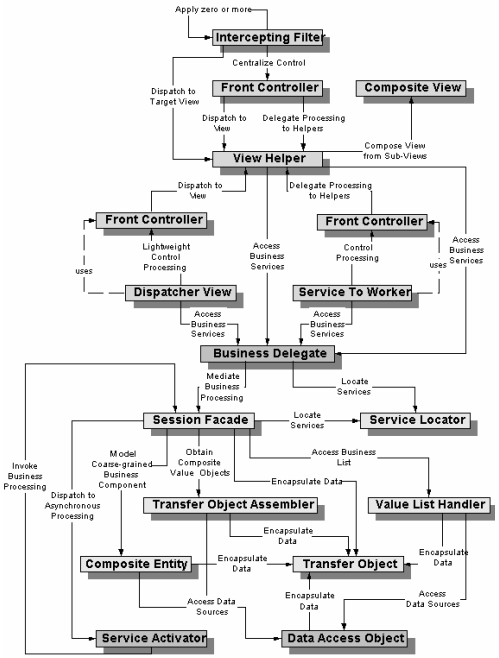
Skladanie podľa potrieb

WEB	Page Flow		Web Flow	
	Struts	Beehive	Spring	JSF Tapestry
Aplikačná logika	Beehive Controls	Spring Beans		
Perzistencia	Spring DAO	Hibernate ORM	Beehive Controls	

SOFTEC

Návrhové vzory

- V softvérovom inžinierstve sú návrhové vzory štandarným riešením bežných problémov v softérovom návrhu. Namiesto zreteľa na to ako jednotlivé komponenty pracujú, návrhové vzory dávajú systematický náhľad, ktorý je zameraný na vzory interakcie komponentov. Návrhové vzory teda popisujú abstraktné systémy interakcie medzi triedami, objektami, komponetami a komunikačný tok.



Vzory pre tvorbu FE

- Intercepting Filter
- Front Controller
- View Helper
- Dispatcher View
- Composite View
- Service to Worker
- Business Delegate
- Service Locator

Návrhové vzory aplikačnej logiky

- Session Facade
- Value List Handler
- Transfer Object
- Composite Entity
- Transfer Object Assembler
- Data Access Object (DAO)
- Adapter pattern

Návrhové vzory – iné (O-O)

- Component Factory
- Configuration provider
- Service Providers
- Helper objects

SOFTEC

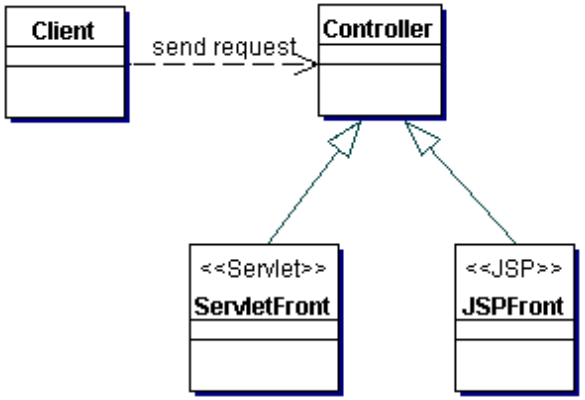
Front Controller

- Centralizácia spoločných systémových služby sa vykonávajú po každý request.
- Spracovanie sa riadi lepšie centralizovaným spôsobom ako replikáciou kódu v mnohých view.*
- Rozhodovacie body sa vytvárajú s ohľadom na získanie a manipuláciu s dátami.
- Viacero views sa používa na zobrazenie podobných aplikačných požiadaviek.
- Centralizovaný bod pre obsluhu požiadaviek je vhodný na riadenie, logovanie používateľských aktivít.
- Umožnenie zložitého/inteligentného riadenia vykonávania a zobrazovania

SOFTEC

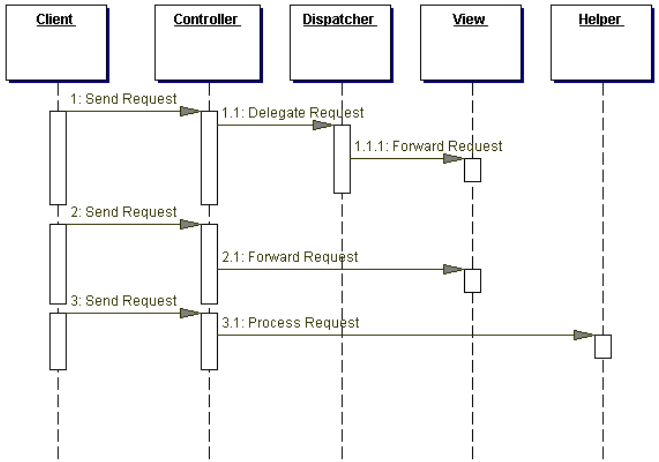
Front Controller

- Štruktúra



SOFTEC

Front Controller



SOFTEC

Front Controller

- Servlet Front Stratégia
  - Kontroler je nezávislý od formátu zobrazovania
  - Vhodné zapuzdrenie riadiacej logiky
  - Prichádza sa však o výhody zmeny riadenia a mechanizmov, ktoré sú prístupné JSP, ale obdobné mechanizmy sú ľahko implementovateľné

SOFTEC

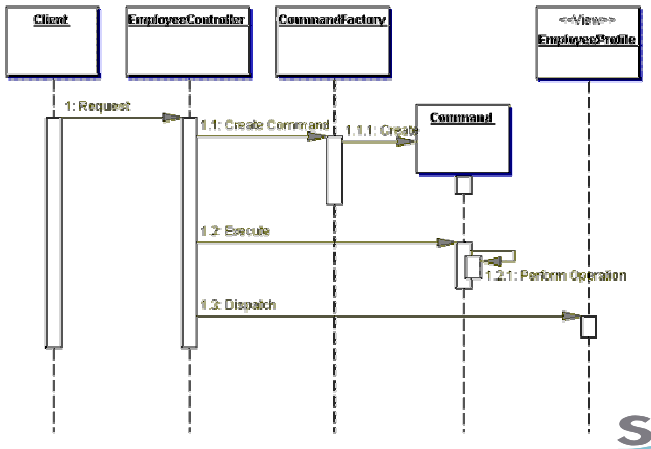
Front Controller

- JSP Front stratégia
  - Je porovnateľné so Servlet stratégiou
  - JSP je však primárne určené pre prezentačný kód
  - Nedá sa rovnako jednoducho vytvárať, kompilovať, testovať a debugovať

SOFTEC

Front Controller

- Command a controller stratégia



SOFTEC

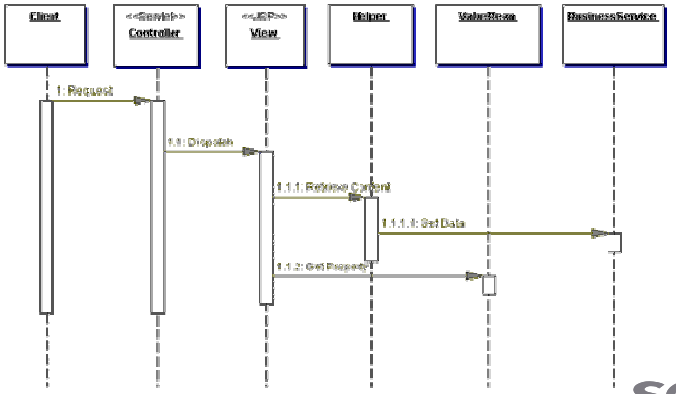
Front Controller

- Resource Mapping stratégia
  - Fyzické mapovanie zdrojov
  - Logické mapovanie zdrojov
  - Násobné mapovanie zdrojov
    - Je logickým mapovaním viacerých zdrojov na jeden fyzický

SOFTEC

Front Controller

- Dispatcher controller stratégia



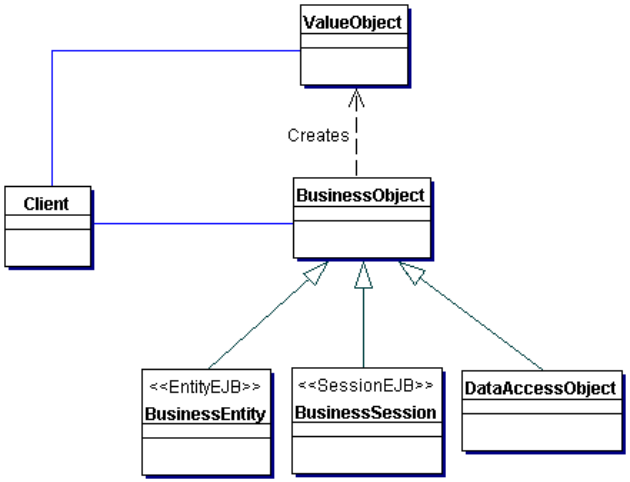
SOFTEC

Transfer Object

- Predpokladá sa, že každý prístup do aplikačnej logiky sa môže vykonávať cez remote interfejsy a preto každé volanie aplikačnej logiky je potenciálne vykonávané cez vzdialené volanie prostredníctvom siete.
- Typické pre aplikácie je, že frekvencia čítania z aplikačných dát je vyššia ako frekvencia ich modifikácie. Klient potrebuje dáta z aplikačnej logiky prezentovať, a vykonávať nad nimi úpravy pre ich zobrazenie. Klient modifikuje dáta vo vrstve aplikačnej logiky, čo sa deje menej často.
- Klient zvyčajne potrebuje hodnoty pre viac atribútov alebo asociovaných objektov z EJB. Tak klient môže invokovať viacero volaní aplikačnej logiky na to, aby obdržal požadované údaje.
- Počet volaní z klienta do aplikačnej logiky má vplyv na zaťaženie siete.

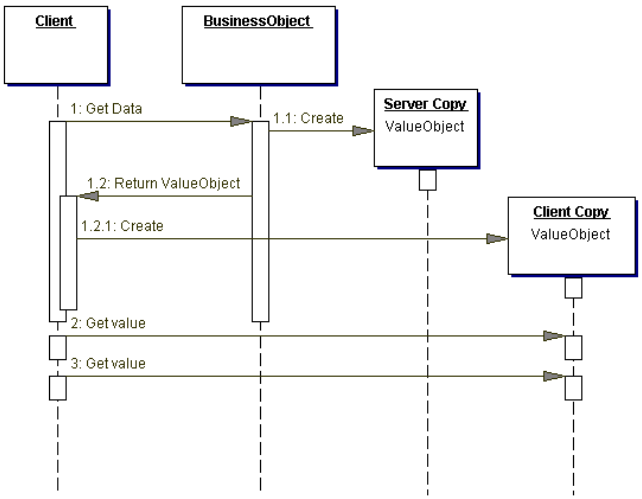
SOFTEC

Transfer Object



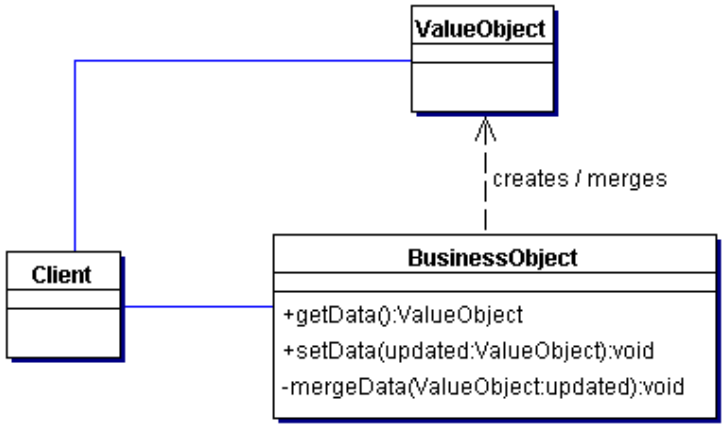
SOFTEC

Transfer Object



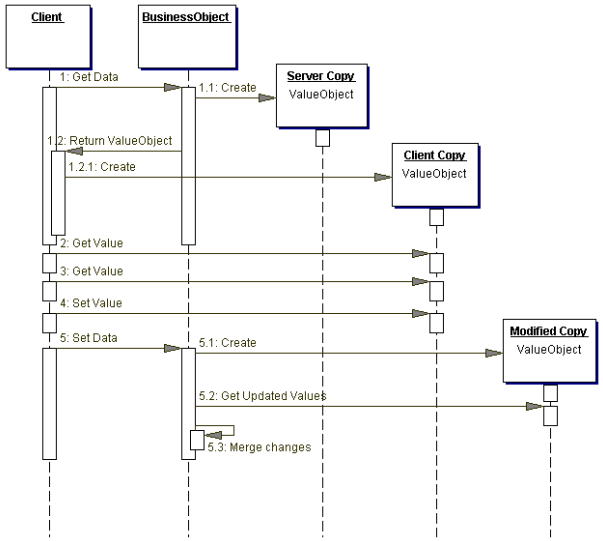
SOFTEC

Transfer Object



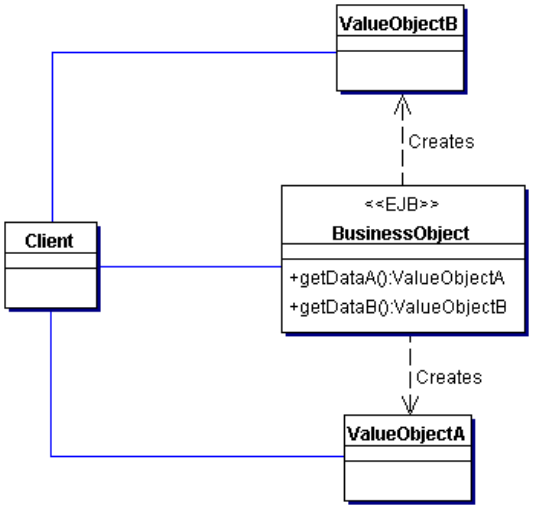
SOFTEC

Transfer Object



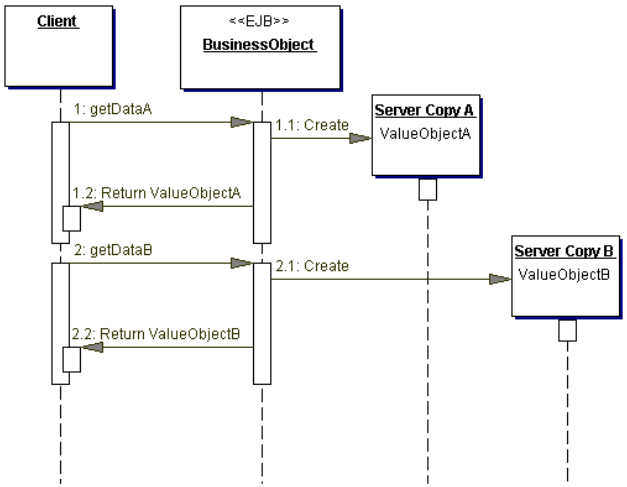
SOFTEC

Multiple Transfer Object



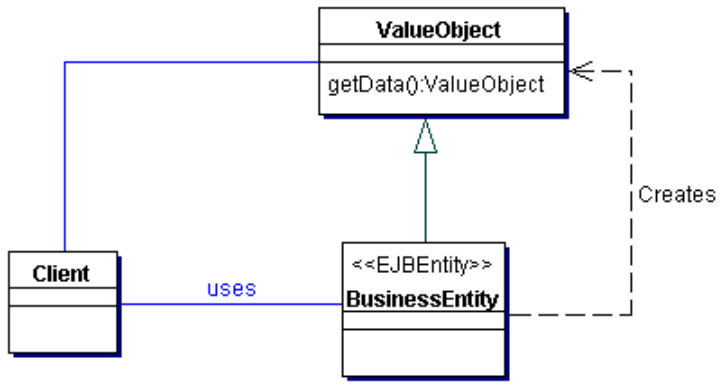
SOFTEC

Multiple Transfer Object



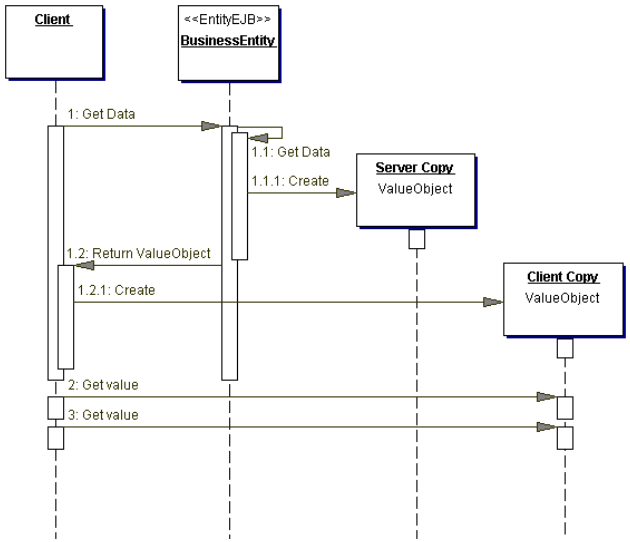
SOFTEC

EJB dedí z VO



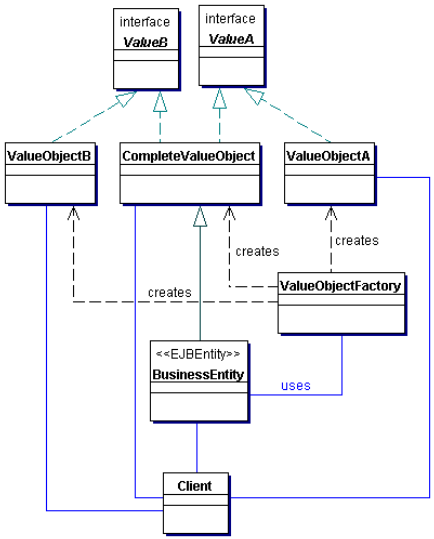
SOFTEC

EJB dedí z VO



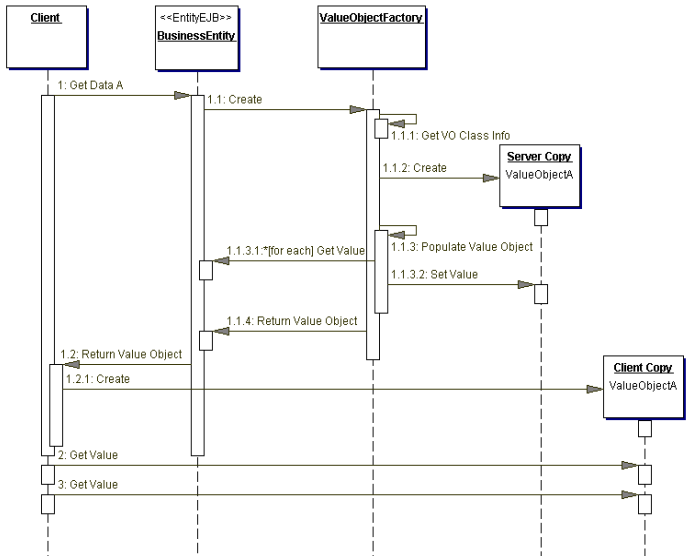
SOFTEC

Transfer Object Factory



SOFTEC

Transfer Object Factory



SOFTEC

Transfer Object - prínosy

- **Zjednodušuje Entity Bean a RemoteInterface**
- **Prenáša viac dát s menším naraz**
- **Znižuje počet volaní aplikačnej logiky**
- **Redukuje sieťovú komunikáciu**
- **Znižuje duplikáciu kódu**
- **Môže zvýšiť komplexicitu kvôli synchronizácii, verziovaniu, riadeniu súčasného prístupu a transakčnosti spracovania**

SOFTEC

Aplikačný Framewok

- Frameworky sú viacnásobne použiteľné miniarchitektúry, ktoré poskytujú všeobecne použiteľnú štruktúru a vlastnosti pre určitú programovú abstrakciu spolu s opisom kontextu použitia a interakcie s okolím v tej ktorej doménovej oblasti. Frameworky obsahujú natvrdo zakomponovaný kontext do podoby virtuálneho stroja, zatiaľ čo abstrakcia, modifikovateľnosť, rozšíriteľnosť a prispôsobivosť je zabezpečená pomocou vstupno-výstupných bodov. Tieto body sú poväčšine realizované pomocou polymorfizmu, dedenia a spätných volaní. Umožňujú prispôsobivosť a rozšíriteľnosť frameworku rôznorodým požiadavkám a podporujú kombinovateľnosť s inými frameworkami. Framework nie je plne funkčná aplikácia, väčšinou je nutné doplniť aplikačnú logiku. Framework riadi interakciu medzi abstraktnými komponentmi s voľnou implementáciou.

SOFTEC

Ako budovať FW

- Štrukturalizácia web vrstvy
  - Komponenty
    - Filtre
    - Servlety
    - JSP
    - Helper triedy
  - Skripty
  - Custom JSP tag libraries
  - Validačné triedy/JS libraries
  - CSS
  - Html templejty

SOFTEC

Ako budovať FW

- Zabezpečenie riadenia toku
  - vychádza sa zo všeobecne uznávaného MVC patternu, kde sú oddelené 3 vrstvy, pričom tok riadi iba jedna cetrálna komponenta
    - Controller (teda nie ze každý formulár robí submit na iný servlet). Riadenie toku robi contoller na zaklade "signálov" alebo dát z iných vrstiev aplikácie - modelu, volania BL, Chybových stavov

SOFTEC

Ako budovať FW

- Mapovanie udalostí (events) na akcie (business requests)
  - Závisí od implementácie interakcie prezentačnej a aplikačnej vrstvy
  - Pokiaľ možno – implementovať konfiguráciou, ktorú je možné počas behu meniť a reloadovať

SOFTEC

Ako budovať FW

- Dynamické toky aplikačných stránok
  - Závisí na zložitosti prezentačných scenárov - webflow a organizácie contolera.
  - Vo všeobecnosti je nutné vytvárať subflow, ktorý má všeobecný interface predávania výsledkov. Najjednoduchší je mechanizmus subflowov organizovat podobne ako volanie funkcií - v nejakom stave aplikácie je vyvolaný subflow a mechanizmus FW sa postará, že tok aplikácie nevybočí nekontroplovaným spôsobom a flow končí v presne plánovaných bodoch.

SOFTEC

Ako budovať FW

- Podpora viacerých typov klientov
  - jedná sa iba o kustomizáciu view vrsvty. Väčšinou to ide ľahko spraviť, ak je view vrstva oddelená od modelu a kontrolera. Na základe identifikácie typu klienta sa rozhodne, aká view vrsvta sa použije (JSP pre browser, JSP pre PDA (musí počítať s malým rozlíšením obrazovky), WML, XML a pod)

SOFTEC

Ako budovať FW

- Použitie šablón/vzorov (templates)
  - Autonómne časti stránok
    - Header
    - Footer
    - Menu
    - Content
  - Sada samostatných komponent
    - Skladanie komponetov na základe grafického vzhľadu a firemnej identite

SOFTEC

Ako budovať FW

- Oddelenie vrstiev
  - Kedy uvažovať o oddelení vrstiev
    - Viacero typov klientov?
    - Zdieľanie business procesov
    - Internet a Intranet prezentácia
    - Integračný charakter aplikácie
    - Koordinácia viacerých zdrojov
  - Prostriedky na realizáciu
    - Použitie návrhových vzorov
    - Vhodný packaging
    - Použitie princípov SOA

SOFTEC

Ako budovať FW

- Riadenie stavu
  - Uloženie stavu
    - HttpRequest
    - HttpSession
    - Globálne Cache
  - Implementácia kontrolera
    - Viacero rozhodovacích bodov
    - Ošetrenie stavov a udalostí na úrovni kontrolera

SOFTEC

Ako budovať FW

- Riadenie „Session“
  - Technicky - web.xml
    - Životnosť
    - Veľkosť
    - Počet
  - Manipulácia prostredníctvom centrálneho mechanizmu
    - Aplikačné dáta
    - Kontextové dáta
    - Dočasné dáta

SOFTEC

Ako budovať FW

- Zdieľanie dát
  - Session pre aktuálneho používateľa
  - Zdieľanie "celoaplikačných“ dát
    - Cez singleton (aj keď v J2EE je tento vzor neodporúčaný, takmer zakázaný)
    - Cez komponent „sediaci“ v niektorom z kontajnerov a plniaci úlohu aplikačného kontextu(Statefull session-EJB, Servlet)

SOFTEC

Ako budovať FW

- Použitie „Cookie“
  - Malý dátový objekt, ktorý je zasielaný z Web servera do web prehliadača a ukladá sa na klientskej stanici. Cookie sa posiela späť s každým requestom z prehliadača.
    - Session ID
    - Používateľské nastavenia
    - Štatistické údaje

SOFTEC

Zhrnutie II.

- Aplikačné scenáre
- MVC návrhový vzor
- Aplikačné návrhové vzory
- Aplikačné frameworky
- Odporúčania

SOFTEC