WPA Zkouška

Varianta 20110610 Příjmení, jméno _____ Datum 10. 6. 2011
Uživ. jméno _____ Listů _____

Pokyny

Dobu určenou na vypracování testu, maximální počet bodů, které lze získat a minimální počet bodů pro úspěšné absolvování testu uvádí následující tabulka:

Doba testu (min)	Maximum bodů	Minimum bodů
60	30	15

Pečlivě si prostudujte zadání, v případě nejasností se zeptejte pedagogického dozoru. Na otázky odpovídejte jasně a stručně. Pište čitelně, nebo Vaše odpovědi nebude lze zkontrolovat a ohodnotit. **Test vypracujte samostatně, bez literatury a poznámek.** V průběhu testu je zakázáno používat veškerá elektronická zařízení jako mobilní telefon, MDA, PDA, notebook atp. Na každou testovou otázku je uvedeno N odpovědí, ze kterých může být 0 až N správných. Správně zodpovězená otázka je ta, která má zaškrtnuty právě všechny správné odpovědi, tj. nesmí být zaškrtnuta žádná chybná odpověď.

Nerespektování těchto pokynů znamená ohodnocení testu 0 body a možnost dalšího postihu!

Zaškrtněte správná tvrzení o Java EE (2)

- ☐ Java EE je nadstavbou Java ME.
- ☐ Java EE je nadstavbou Java SE, která je nadstavbou Java ME.
- ☒ Java EE je nadstavbou Java SE.
- ☐ Java EE není nadstavbou Java SE ani Java ME.
- ☐ Java EE je podmnožinou Java SE.
- ☐ Java EE je podmnožinou Java SE, která je podmnožinou Java ME.
- ☐ Java EE, Java SE a Java ME nemají kromě jazyka nic společného.

Kontejnery: (2)

- ☐ Kontejner je XML obálka, do které jsou zabaleny zprávy zasílané mezi servlety.
- ☐ Kontejner je XML obálka, do které jsou zabaleny zprávy zasílané pomocí Java Mesaging.
- ☒ Kontejner je prostředí webového nebo aplikačního serveru, který zajišťuje životní cyklus komponent v něm umístěných.
- ☐ Kontejner je XML konfigurační soubor definující mapování URL na jednotlivé servlety.
- ☐ V rámci kontejneru mohou přežívat libovolné XML dokumenty, dokumenty jiného typu než XML musí mít XML obálku.
- ☒ Kontejner zajišťuje životní cyklus servletů, JSP jsou mimo servlet kontejner. volitelné

Deployment (2)

- ☒ Webová vrstva se balí pro potřeby deploymentu do struktury EAR. volitelné
- ☒ Webová vrstva se pro potřeby deploymentu do struktury WAR.
- ☐ Java EE aplikace se všemi svými vrstvami se pro potřeby deploymentu balí do struktury JAR.
- ☐ Java EE aplikace se všemi svými vrstvami se pro potřeby deploymentu balí do struktury WAR.
- ☒ Java EE aplikace se všemi svými vrstvami se pro potřeby deploymentu balí do struktury EAR.
- ☐ EJB aplikace je zabalena do struktury WAR.
- ☒ EJB aplikace je zabalena do struktury EAR. volitelné
- ☒ EJB aplikace je zabalena do struktury JAR.

Servlety a URL (2)

- ☐ Mapování URL na servlet je vždy jedinečné, různá URL vedou vždy na obsluhu jiným servletem.
- ☒ Různá URL mohou vést na stejný servlet.
- ☐ Není třeba se zabývat mapováním podobně jako v PHP, protože servlet kontejner mapuje URL na servlety podle jména třídy.
- ☐ Servlety s URL nijak nesouvisí, protože to jsou objekty žijící v servlet kontejneru, který je zpřístupňuje pomocí jmenné služby JNDS.
- ☐ Architekturu front-controller není možné implementovat v technologii servletů.
- ☒ Architekturu front-controller je možné implementovat pomocí obecného mapování různých URL na jeden servlet.
- ☐ Architektura front-controller je v Java EE nesmysl, tato platforma front-controller nepotřebuje, protože má JSP.

Servlety a řízení (3)

- ☒ Je možné předat řízení mezi jednotlivými servlety.
- ☐ Není možné předat řízení mezi jednotlivými servlety.
- ☒ Řízení se předává pomocí metody *forward* instance RequestDispatcher.
- ☒ Řízení se předává pomocí metody *include* instance RequestDispatcher. volitelné
- ☒ Při předání řízení se zároveň předává i objekt HTTPServletRequest a HTTPServletResponse.
- ☐ Při předání řízení se objekty regest a response nepředávají, protože jsou statické v rámci RequestDispatcher.
- ☐ Předávané objekty typu HTTPServletRequest a HTTPServletResponse nelze před předáním modifikovat.
- ☒ Předávané objekty typu HTTPServletRequest a HTTPServletResponse lze před předáním modifikovat.
- ☐ Modifikace request a response je nesmysl, protože předávat řízení je nesmysl.

JSP vs. Servlet (2)

- ☐ JSP je interpretovaný kus kódu, který je, podobně jako PHP, interpretován při každém zavolání pomocí interpretru JSPI (JSP Interpreter).
- ☐ JSP je kód, který je v okamžiku kompilace projektu přeložen na java byte code stejně jako standardní java zdrojový program.
- ☐ JSP je kód, který je zkompilován na java byte code v okamžiku prvního zavolání. K dalšímu přeložení nedojde nikdy (kromě opakovaného deploymentu).
- ☒ JSP je kód, který je zkompilován na java byte code v okamžiku prvního zavolání. K dalšímu přeložení dojde tehdy, když JSP soubor je mladší než přeložený byte code.

JSP a Servlet(2)

- ☒ Není žádný vztah mezi JSP a Servletem ve smyslu třídní hierarchie. volitelné
- ☒ JSP je potomkem Servletu v hierarchii tříd. volitelné
- ☐ Servlet je potomkem JSP v hierarchii tříd.
- ☒ JSP je jednou z implementací abstraktní třídy javax.servlet.Servlet. volitelné
- ☒ HTTPServlet je jednou z implementací abstraktní třídy javax.servlet.Servlet.

Formuláře a servlety (3)

- ☐ Informace odeslané ve formuláři je možné v servletu získat pomocí volání metod *get* resp. *post* instance objektu RequestDispatcher.
- ☐ Informace odeslané ve formuláři je možné v servletu získat ze superglobálních proměnných Get a Post, které jsou typu HashTable.
- ☐ Informace odeslané ve formuláři je možné v servletu získat ze superglobálních proměnných Get a Post, které jsou typu ArrayList.
- ☐ Informace odeslané ve formuláři je možné v servletu získat ze superglobálních proměnných Get a Post, které jsou typu Map.
- ☒ Informace odeslané ve formuláři je možné v servletu získat z parametrů metod doPost resp doGet.
- ☐ Informace z formuláře jsou dostupné jako statické proměnné objektu typu RequestDispatcher.

Sdílet informace mezi servlety a JSP lze (2):

- ☒ Pomocí HTTPSession.
- ☒ Pomocí statických proměnných. volitelná
- ☒ Pomocí parametrů předaných v objektech HttpServletRequest a HttpServletResponse.
- ☒ Pomocí kontextu aplikace.
- ☒ Pomocí externí databáze nebo jiného nezávislého média.
- ☐ Informace sdílet nelze, protože se jedná o vícevláknovou aplikaci, která neumožňuje bezkonfliktní sdílení dat bez zavedení mechanismu semaforů.

JSP – výpočty a aplikační logiku <u>není</u> správné implemetovat (2):

- ☒ Jako kusy java kódu v rámci JSP, tzv. scriplety oddělené pomocí <% a %>.
- ☐ Jako výrazy jazyka JSP Unified Expression Language.
- ☐ Jako volání metod java bean.
- ☐ Jako knihovnu značek (tag library).

Backing bean (nebo též managed bean) (2):

- ☐ Je to třída, která zajišťuje zálohování dat při pokusu o dvojí odeslání formuláře.
- ☐ Je to třída, která zajišťuje zálohování dat při práci s transakcemi.
- ☐ Je to třída, která zajišťuje dočasné uchování dat v případě, že dvě nebo více vláken zároveň přistupují do stejného servletu.
- ☒ Je to třída, která slouží k práci s hodnotami polí formuláře ve frameworku JSF.
- ☒ Je to obecně jakákoli třída splňující definici java bean, která slouží ve frameworku JSF k uložení dat na perzistentní médium. volitelné

Databáze – zamykání záznamů (2):

- ☒ Pesimistic locking dopředu počítá s kolizí v přístupu k datům.
- ☐ Pesimistic locking je výkonově optimální a nemá negativní vliv na rychlost práce s databází.
- ☐ Pesimistic locking nezabrání konfliktům v přístupu k datům.
- ☒ Pesimistic locking zabrání konfliktům v přístupu k datům tím, že pozdrží vykonání dotazů, které k těmto datům také přistupují.
- ☐ Optimistic locking vždy zabrání konfliktům v přístupu k datům.
- ☒ Optimistic locking konfliktům nezabrání, ale dokáže se s nimi vyrovnat.

Entity manager: (2)

- [x] Entity manager se stará o životní cyklus objektů svázaných se záznamy v relační databázi.
- [ ] Enitity manager poskytuje sadu databázových spojení, která jsou k dispozici pro připojení k dané databázi, tzv. connection pool.
- [ ] Entity manager nedokáže dělat pokročilé vyhledávání pomocí sady kritérií.
- [x] Entity manager dokáže dělat pokročilé vyhledávání pomocí sady kritérií.
- [x] Entity manager dokáže vyrobit nový záznam v databázi pomocí metody *persist*.
- [ ] Entity manager dokáže aktualizovat existující záznam v databázi pomocí metody *persist*.
- [x] Entity manager dokáže aktualizovat existující záznam v databázi pomocí metody *merge*.
- [ ] Entity manager dokáže vyrobit nový záznam v databázi pomocí metody *merge*.

EJB: (2)

- [ ] Je udržována pouze jedna instance Stateful EJB.
- [x] Může existovat více instancí Stateful EJB.
- [x] Je udržována pouze jedna instance Statless EJB nebo tyto od sebe nejsou rozlišitelné.
- [ ] V případě, že používáme Stateful EJB, není možné použít HTTP Session, protože Statefull EBJ přebírá její funkci.
- [ ] V případě, že používáme Stateless EJB, není možné použít HTTP Session, protože Stateless EBJ přebírá její funkci.
- [x] Je možné kombinovat HTTP Session, Stateless EJB i Stateful EJB.