Novinky v JEE 6

Petr Aubrecht
CA

Co dnes probereme

- Novinky v JEE 6
  - základní cíle
  - změny v infrastruktuře
  - webový framework
  - vylepšení EJB

Co už všechno umíte

- Už umíte všechno.
- Jste spokojeni?
- Nešlo by to JEŠTĚ jednodušeji/efektivněji/rychleji?
- Jak vylepšit podporu ze strany nástrojů, serverů, hostingu...

Základní cíle JEE 6

- More Flexible Technology Stack
  - Ne všechny části jsou vždy potřeba (např. CORBA), modulárnost.
- Enhance Extensibility
  - self registration, technologie i frameworky, knihovny
- Further Ease of Development
  - anotace místo web.xml, jednodušší packaging
- Powerful New Technologies
  - JAX-RS (RESTful Web Services)
  - Contexts and Dependency Injection
  - Bean Validation

JAX-RS (RESTful Web Services)

```
@Path ("items")
  @Produces (MediaType.APPLICATION_XML)
  Public class ItemsResource {

     @Context UriInfo uriInfo;

     @GET
     Items listItems() {
        Return Allitems();
     }

     @POST
      @Consumes (MediaType.APPLICATION_XML)
     Public Response create(Item item) throws ItemCreationException {
        Item newItem = createItem(item);
        URI newItemURI = uriInfo.getRequestUriBuilder().path(newItem.getId()).build();
        return Response.created(newItemURI).build();
     }
     ...
  }
```

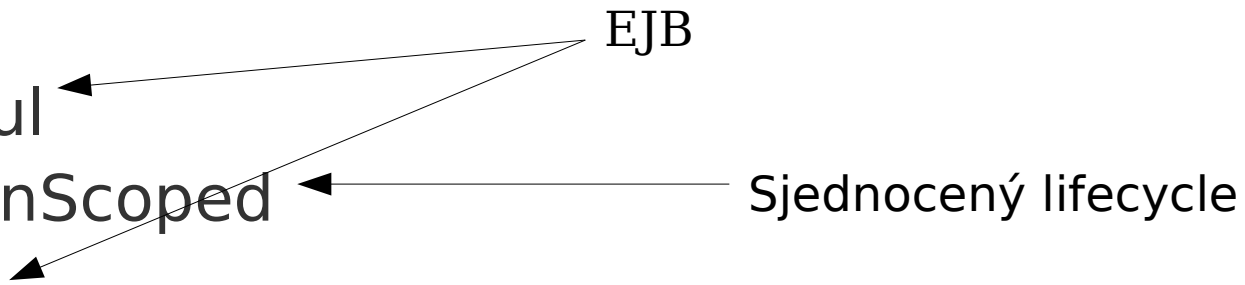
Contexts and Dependency Injection (CDI) (1)

- sjednocení EJB a JSF managed beans
- EJB může nahradit JSF beany

```
@Model
   public class Credentials {
      private String username;
      private String password;
```

- @Model jako model v MVC

CDI (2)




```
@Stateful
@SessionScoped
@Model
public class Login {
    @Inject Credentials credentials;
    @Inject EntityManager userDatabase;

    private User user;

    @TransactionAttribute(REQUIRES_NEW)
    @RolesAllowed("guest")
    public void login() {
      ...
    }
```

CDI (3)

```
@RolesAllowed("user")
@Produces @LoggedIn User getCurrentUser() {
   ...
}
```

- @Inject – inject pro CDI
- @Produces – generátor, použije se v případě, že je daný typ potřeba vytvořit
- lze vytvořit i generátory a konzumenty událostí (events)

Bean Validation

- Validace je umístěna v modelu jako anotace.
- Jakýkoliv framework je může využít – JPA, JSF ap.
- Není třeba tyto informace uvádět v každém frameworku znova.

```
public class Address {
        @NotNull @Size(max=30)
        private String addressline1;
```

- Je možné definovat i vlastní validátory (např. ZIP code).
- Jak se validace používá? Je automatická!

Web Tier Enhancements

- Servlet 3.0
  - Web Fragments
  - Asynchronous Processing
  - Simplified Page Authoring
    - facelets/templates
    - composite components
- Ajax in JSF 2.0
- Other new features
  - adding servlets and filters dynamically
  - programmatic security
  - resource handling (resources directory)

Web Fragments (1)

- web.xml lze rozdělit na více částí
  - web-fragment.xml

```
<web-fragment>
  <servlet>
    <servlet-name>...</servlet-name>
    <servlet-class>...</servlet-class>
  </servlet>
</web-fragment>
```

Web Fragments (2)

- Web fragments dovoluje pořadí fragmentů.
- Zpracování fragmentů lze i potlačit (<metadata-complete/>).
- Konečně se frameworky mohou samy registrovat!
- Webové frameworky nepotřebují definovat front controller, definují si ho samy.

Shared Framework Pluggability

- Kontejner prochází třídy a vyhledává všechny implementace rozhraní ServletContainerInitializer.
- Knihovny se tak mohou registrovat i programově.

```
@HandlesTypes(AnnotationA.class)
AServletContainerInitializer implements ServletContainerInitializer {
    public void onStartup(Set<Class<A>>c, ServletContext ctx)
            throws ServletException {
  // Framework-specific code here to initialize the runtime
  // and setup the mapping etc.
  ServletRegistration reg = ctx.addServlet("AServlet",
                                           "com.f.AServlet");
  reg.addServletMapping("/foo");
```

Asynchronous Processing in Servlet 3.0

- Jednoduché řešení dlouhotrvajících requestů bez blokování zpracujících threadů, případně pushing (server to client)

```
@WebServlet(name="CalculatorServlet", asyncSupported=true,
urlPatterns={"/calc", "/getVal"})

  public class CalculatorServlet extends HttpServlet{

    public void doGet(HttpServletRequest req, HttpServletResponse
res) {
      AsyncContext aCtx = req.startAsync(req, res);
      ...
    }
    ...
  }
```

Composite Components

- Často je potřeba custom component, ale není třeba speciální vykreslování, jen složení již existujících.
- JSF 2.0 podporuje tzv. composite component, která přesně toto definuje. Příkladem budiž label a edit společně s chybovou (validační) hláškou.
- V komponentě se definují akce, které jsou posléze využívány pomocí listenerů.

```
<h:body>
   <composite:interface>
   <composite:actionSource name="loginEvent"/>
   </composite:interface>
   <composite:implementation>
    <p>Username:<h:inputText id="username" /></p>
```

```
<ez:loginPanel>
<f:actionListener
    for="loginEvent"
    type="LoginListener"
 />
```

Support for Ajax in JSF 2.0

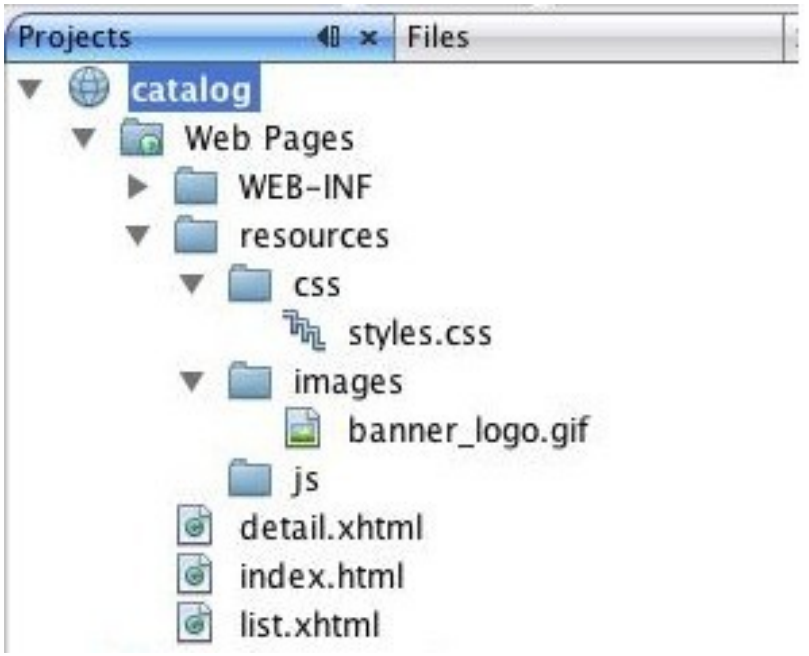
- Hlavní myšlenkou jsou částečné updaty, kdy je změněna pouze část stránky. V RichFaces byla tato funkcionalita poskytována knihovnou a4j.

```
<h:commandButton id="button1">
    <f:ajax execute="..." render="..."/>
</h:commandButton>
```

- Tento způsob má několiv velice výhodných vlastností:
  - rychlejší odezva (generuje se pouze malá část stránky)
  - lokální změna (např. zůstanou vybrané položky v listech, selekce textu ap.)

Resource Handling

- Zdroje jako css, obrázky, Javascript jsou nyní standardně umístěny v adresáři resources.
- Existuje k nim i nový interface.
- Pro aplikaci to nemá žádnou výraznou výhodu, ale pro knihovny je to výrazný pokrok.



Zjednodušení EJB – Even Easier to Use

- Přesně ve stylu „further ease of development“ - žádná revoluce, pouze usnadnění některých situací, které se dosud řešily obtížně a nesystémově.

- No-interface View
- Singleton
- Asynchronous Session Bean Invocation

No-interface View

```
@Stateless
   public class HelloBean {
```

- Nyní není třeba používat rozhraní.
- V případě lokálního rozhraní stejně nemá velký význam.
- Klient samozřejmě nesmí použít new, reference se dělá normálně pomocí @EJB.

Singleton

- Nový typ session beany, který je v celé aplikaci pouze jednou.

```
@Singleton
   public class PropertiesBean {
```

- Dostupné jsou jako obvykle – přes @EJB.
- Metody jsou thread-safe, takže kontejner zajistí, že nejsou volány z více threadů najednou.
- Bohužel, není definováno chování v clusteru, pouze v rámci jedné JVM.

Asynchronous Session Bean Invocation

```
@Asynchronous
public java.util.concurrency.Future<Integer> performCalc(...) {
        // ... do calculation
        Integer result = ...;
        return new AsyncResult<Integer>(result);
}
```

- Metody mohou vracet výsledky asynchronně.
- Další podpora dlouhotrvajících operací.

Simplified Packaging, EJB

- Není potřeba balit jar a war do ear, stačí pouze war.
- EJB Lite je podmnožina EJB 3.1, která stačí ve většině případů (zapadá do kontextu profilů).
- EJB 3.1 mají nyní Embeddable API, které dovoluje iniciovat EJB i v desktopových aplikacích (vhodné především pro testování).
- Anotace @Startup zařídí, že singleton je vytvořen při startu aplikace, takže @PostConstruct je volán na začátku.

Vylepšení JPA 2.0

- Sorted lists – pomocí @OrderBy lze říct, jak mají být objekty v kolekcích (@OneToMany) uspořádány.
- Element Collection:

public **enum FeatureType** { AC, CRUISE, PWR, BLUETOOTH, TV, ... }

```
@Entity public class Vehicle {
        @ElementCollection
        @CollectionTable(name="VEH_OPTNS")
        @Column(name="FEAT")
        Set<FeatureType> optionalFeatures;
```

EJB QL

- CASE, NULLIF, COALESCE
  - práce s NULL ve sloupci
- Criteria API
  - vytváření dotazů dynamicky (programovaný složitější SELECT)
- Support for pessimistic locking (em.lock(obj1, READ))
  - vhodné pro případy, kdy dochází často ke kolizím
  - 6 režimů (3 optimistické, 3 pesimistické)
- <validation-mode> v persistence.xml určuje, jak budou využívány validace

Profily

- Již nyní je JEE opravdu široká technologie s mnoha požadavky, které většina aplikací ani nevyužije (viz oblíbený terč CORBA)
- Pro JEE 6 je navržena myšlenka profilů, kdy kontejner musí implementovat určité technologie; není potřeba implementovat úplně všechny.
- Aktuálně je definován pouze Web Profile, který obsahuje webové technologie, EJB mimo JMS a JavaMail, ale nemusí obsahovat web services a management a security.
- Objevují se nové technologie, které budou později začleněny jako nepovinné (např. SIP Servlety, JAX-RS).

Co dál

- Naučit se všechny části JEE opravdu dobře je mimo schopnosti jednoho člověka. Ujasněte si, která část vás baví a tu se naučte po hloubky.
- Časem se dostanete do pozice architekta nebo manažera a budete o nasazení těchto technologií rozhodovat. Měli byste vědět o jejich výhodách i slabinách!
- O praktickém nasazení se dozvíte v posledních třech přednáškách.

Links

- http://java.sun.com/javaee/
- http://java.sun.com/developer/technicalArticles/JavaEE/JavaEE6Overview.html
- http://java.sun.com/javaee/javaserverfaces/
- http://download-book.net/jsf-2.0-book-pdf.html
- Beginning Java EE 6 with GlassFish 3
- JavaServer Faces 2.0, The Complete Reference