Transaction API
+ webové frameworky
+ další poznámky

Petr Aubrecht
CA

Co dnes probereme

- Slíbené JTA
- Webové frameworky (serverové)
- Další
  - Třívrstvá architektura
  - Různé z JEE 6
  - Doporučení pro postup práce v JEE

Java Transaction API (JTA)

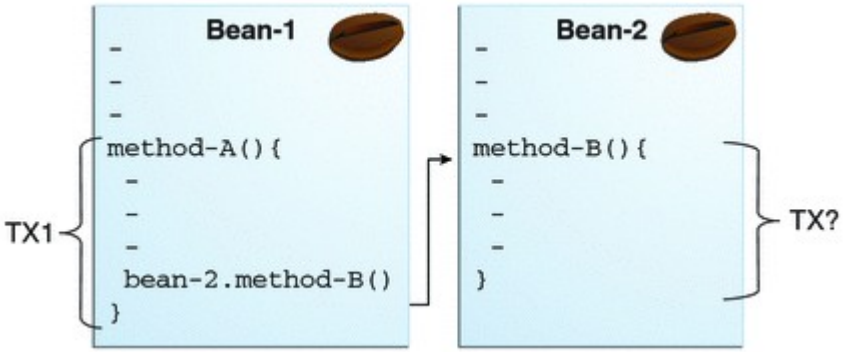
- Něco definice:
  - Java Transaction API (JTA) specifies standard Java interfaces between a transaction manager and the parties involved in a distributed transaction system: the resource manager, the application server, and the transactional applications.
  - The JTA specification was developed by Sun Microsystems in cooperation with leading industry partners in the transaction processing and database system arena. See JSR 907. Also see the Java Transaction Service (JTS) page

Bean Managed Transaction

- javax.transaction.UserTransaction interface
  - begin()
  - commit()
  - rollback()

Container Managed Transactions

- Probrali jsme u session bean, default nastavení je @Required, tedy transakce je vždy použita pro každou metodu (session beany)

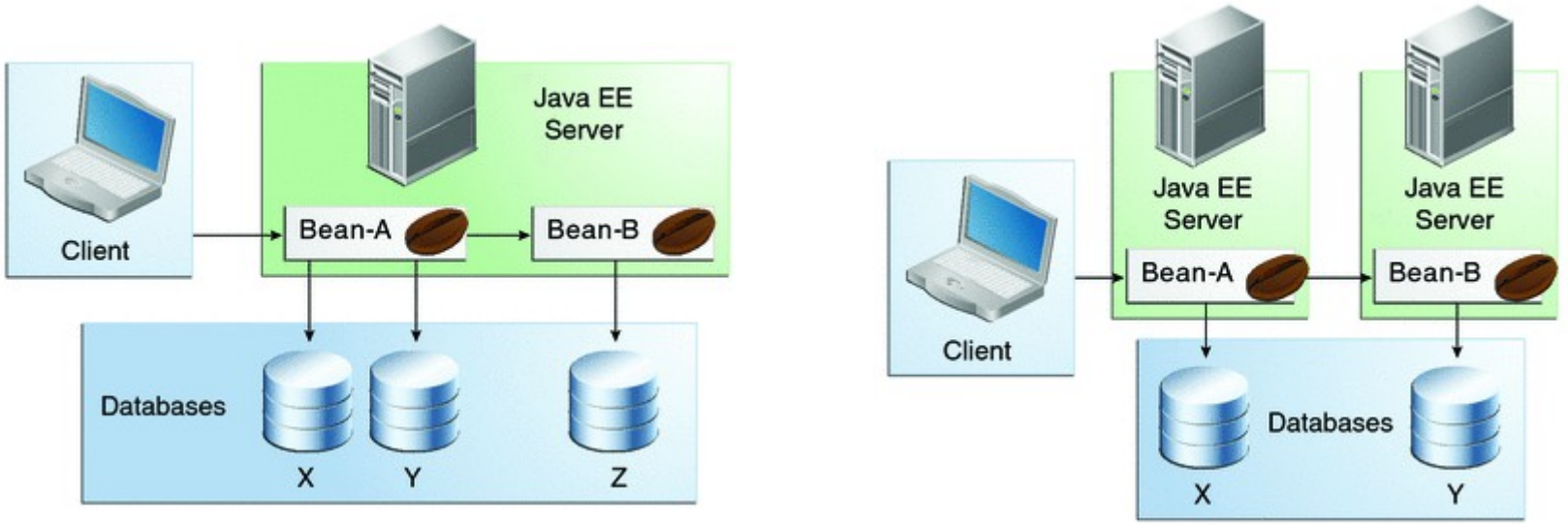


Attributes

Transaction Attr.	Client's Tr.	Business Method's Tr.
Required	None	T2
Required	T1	T1
RequiresNew	None	T2
RequiresNew	T1	T2
Mandatory	None	Error
Mandatory	T1	T1
NotSupported	None	None
NotSupported	T1	None
Supports	None	None
Supports	T1	T1
Never	None	None
Never	T1	Error

JTA – toť vše

- Je to samozřejmě složitější (např. synchronizaci napříč servery), ale nad rámec našich potřeb.



Webové frameworky

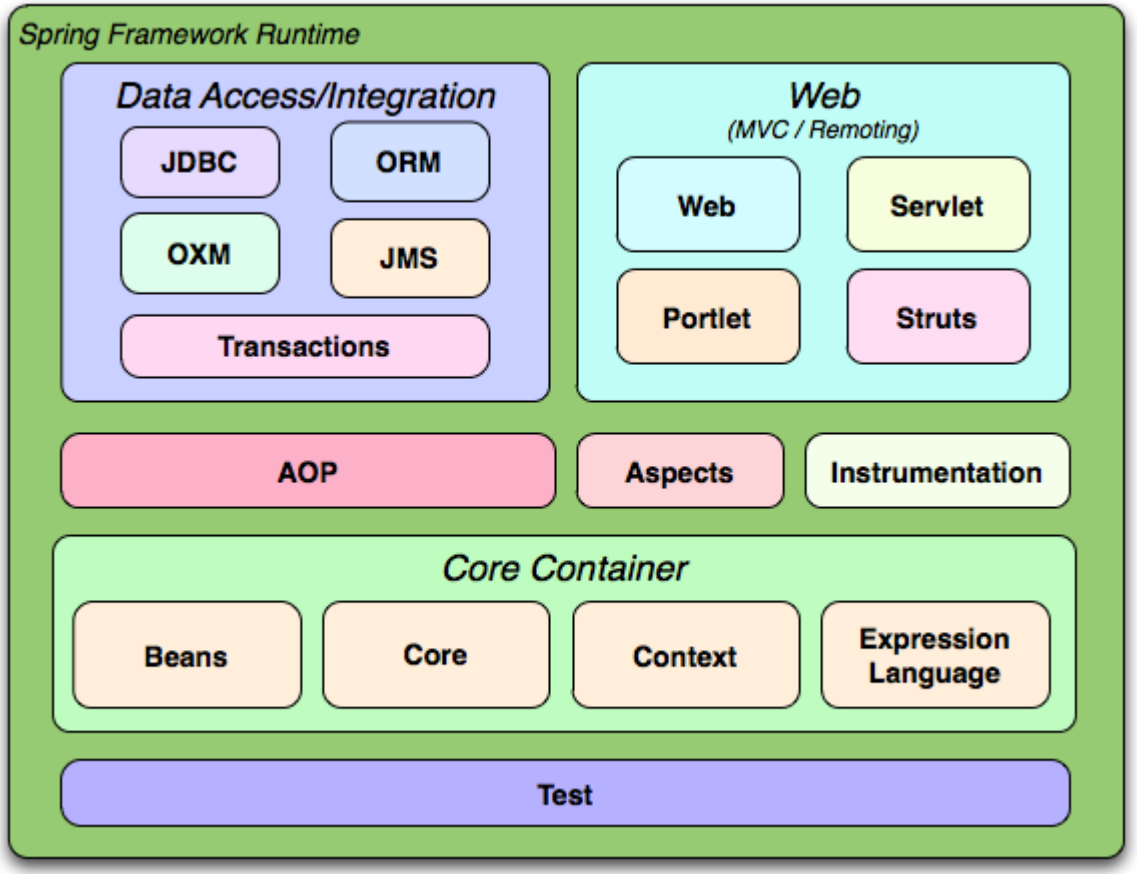
- Čistě Javascriptové knihovny necháme na příští přednášku Martina Ptáčka.
- Další možností je použití **cloudu (PaaS)**
  - např. Amazon (v případě zájmu řekněte Martinovi), dostanete k dispozici celý server, kam můžete dát, co libo
  - Google App Engine – neumí JEE, tedy session beany, má vlastní API, JPA 1.0+JDO, umí Java, Python, Go
- Zabývat se budeme především frameworky nad Javou nebo aspoň JVM (o MS se dozvíte v jiných předmětech).
- RIA – Rich Internet Application

Hlavní rozdělení – client/server

- Serverové frameworky řeší především procesy na serveru, tedy IoC, transakce ap.
  - **JEE – session beans, JPA**
  - Spring (+DataServices, WebFlow)
- Klientské
  - komponentové – nabízejí grafické komponenty: GWT (Vaadin), **JSF**
  - nekomponentové – pomáhají s HTML: Spring MVC, Grails (RoR), Wicket

Spring Framework

- Spring vznikl jako knihovna pro jednodušší programování JEE při psaní knihy Expert One-on-One J2EE Design and Development Roda Johnsona, později se přetvořila na samostatnou knihovnu konkurující JEE, fungující jako integrační platforma pro mnoho technologií.
- Spring v mnoha aplikacích slouží na backendu jako IoC, tedy stará se o inicializaci tříd.



Spring IoC, DI

```
<!-- property-based -->
<bean id="exampleBean" class="examples.ExampleBean">
<!-- setter injection using the nested <ref/> element -->
<property name="beanOne"><ref
bean="anotherExampleBean"/></property>
<!-- setter injection using the neater 'ref' attribute -->
<property name="beanTwo" ref="yetAnotherBean"/>
<property name="integerProperty" value="1"/>
</bean>
<bean id="anotherExampleBean" class="examples.AnotherBean"/>
<bean id="yetAnotherBean" class="examples.YetAnotherBean"/>
```

```
<!-- constructor-based -->
<bean id="exampleBean" class="examples.ExampleBean">
<constructor-arg index="0" value="7500000"/>
<constructor-arg index="1" value="42"/>
</bean>
```

Spring Web Flow

- Spring Web Flow is a Spring MVC extension that allows implementing the "flows" of a web application. A flow encapsulates a sequence of steps that guide a user through the execution of some business task. It spans multiple HTTP requests, has state, deals with transactional data, is reusable, and may be dynamic and long-running in nature.

Spring Data

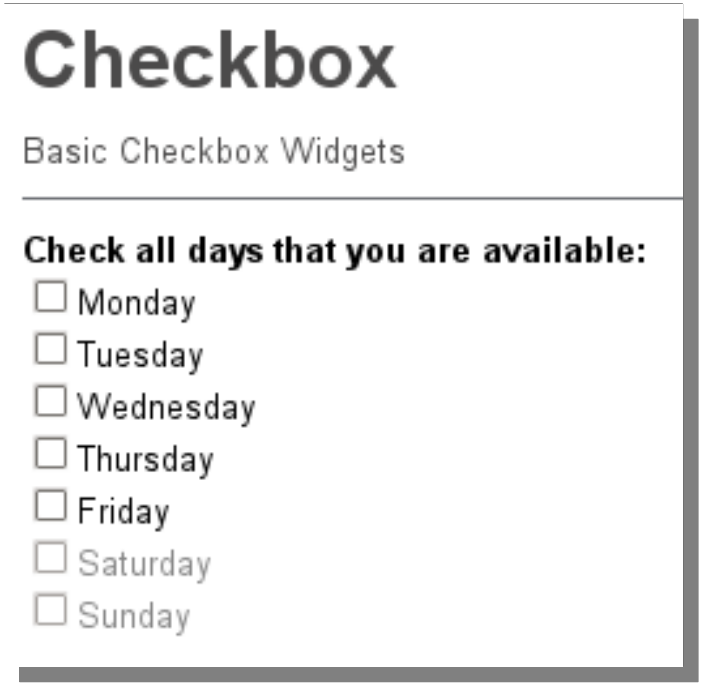
Category	Sub-project
Relational Databases	JPA
	JDBC Extensions
Big Data	Apache Hadoop
Data-Grid	GemFire
Key Value Stores	Redis
	Riak
Document Stores	MongoDB
	CouchDB (planned)
Graph Databases	Neo4j
Column Stores	HBase (planned)
	Cassandra (planned)
Blob-Stores	Blob
Common Infrastructure	Commons
	Grails Mapping

GWT

- Google Web Toolkit
- Hlavní idea: programovat UI v Javě, která se potom přeloží do Javascriptu
- GWT v sobě nese kompilátor Javy, jehož výstup překládá do Javascriptu, pro každý browser zvlášť.
- Zdrojový kód smí obsahovat importy pouze z java.lang, java.util a z GWT knihoven.
- Stránka je vytvářena procedurálně, kódem, nikoliv XML definicí, velmi podobně jako Swing.
- GWT řeší komunikaci client-server asynchronně.

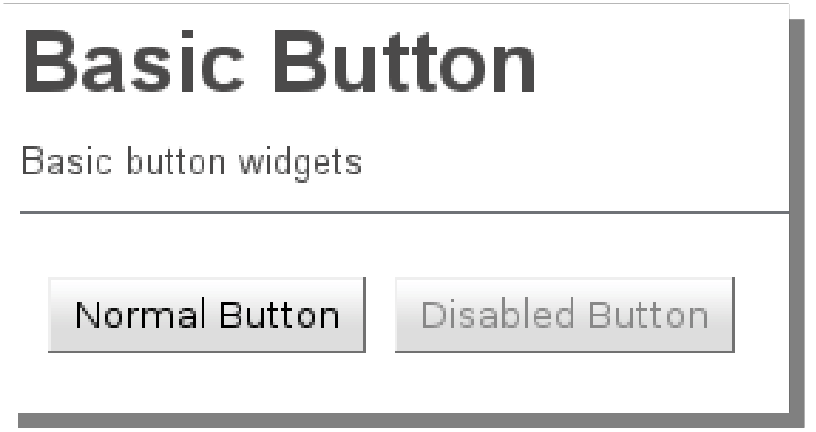
Příklad GWT – panel

```
public Widget onInitialize() {
  VerticalPanel vPanel = new VerticalPanel();
  HTML label = new HTML(constants.cwCheckBoxCheckAll());
  vPanel.add(label);
  // Add a checkbox for each day of the week
  String[] daysOfWeek = constants.cwCheckBoxDays();
  for (int i = 0; i < daysOfWeek.length; i++) {
    String day = daysOfWeek[i];
    CheckBox checkBox = new CheckBox(day);
    checkBox.ensureDebugId("cwCheckBox-" + day);
    // Disable the weekends
    if (i >= 5) {
      checkBox.setEnabled(false);
    }
    vPanel.add(checkBox);
  }
  // Return the panel of checkboxes
  return vPanel;
```



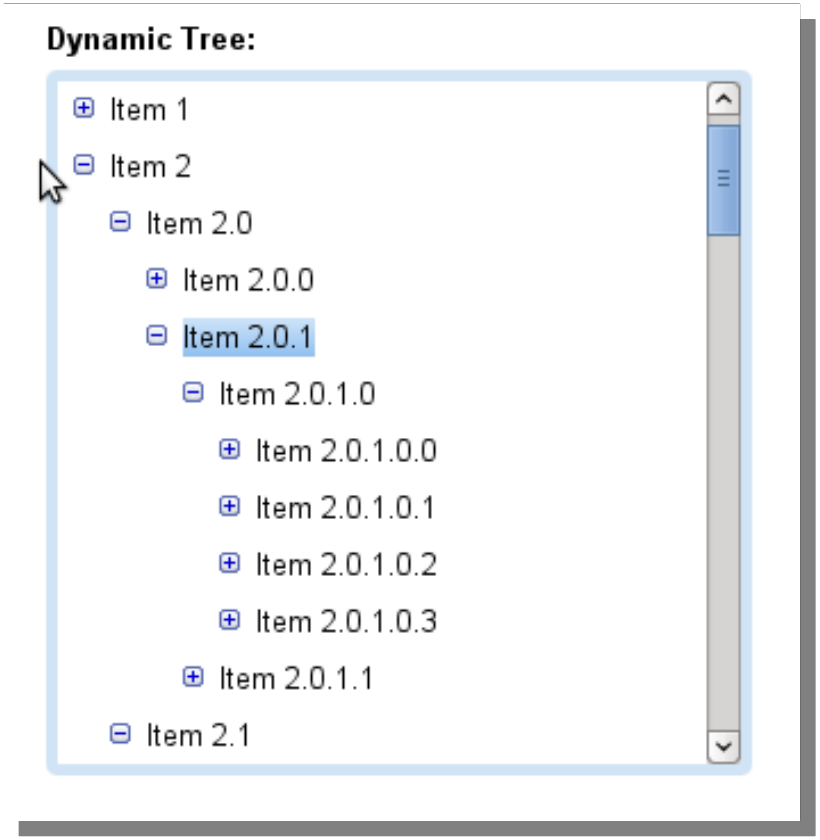
Příklad GWT – akce

```
public Widget onInitialize() {
  // Create a panel to align the widgets
  HorizontalPanel hPanel = new HorizontalPanel();
  // Add a normal button
  Button normalButton = new Button(
      constants.cwBasicButtonNormal(), new ClickHandler() {
        public void onClick(ClickEvent event) {
          Window.alert(constants.cwBasicButtonClickMessage());
        }
      });
  hPanel.add(normalButton);
  // Add a disabled button
  Button disabledButton =
      new Button(constants.cwBasicButtonDisabled());
  disabledButton.setEnabled(false);
  hPanel.add(disabledButton);
  return hPanel;
}
```

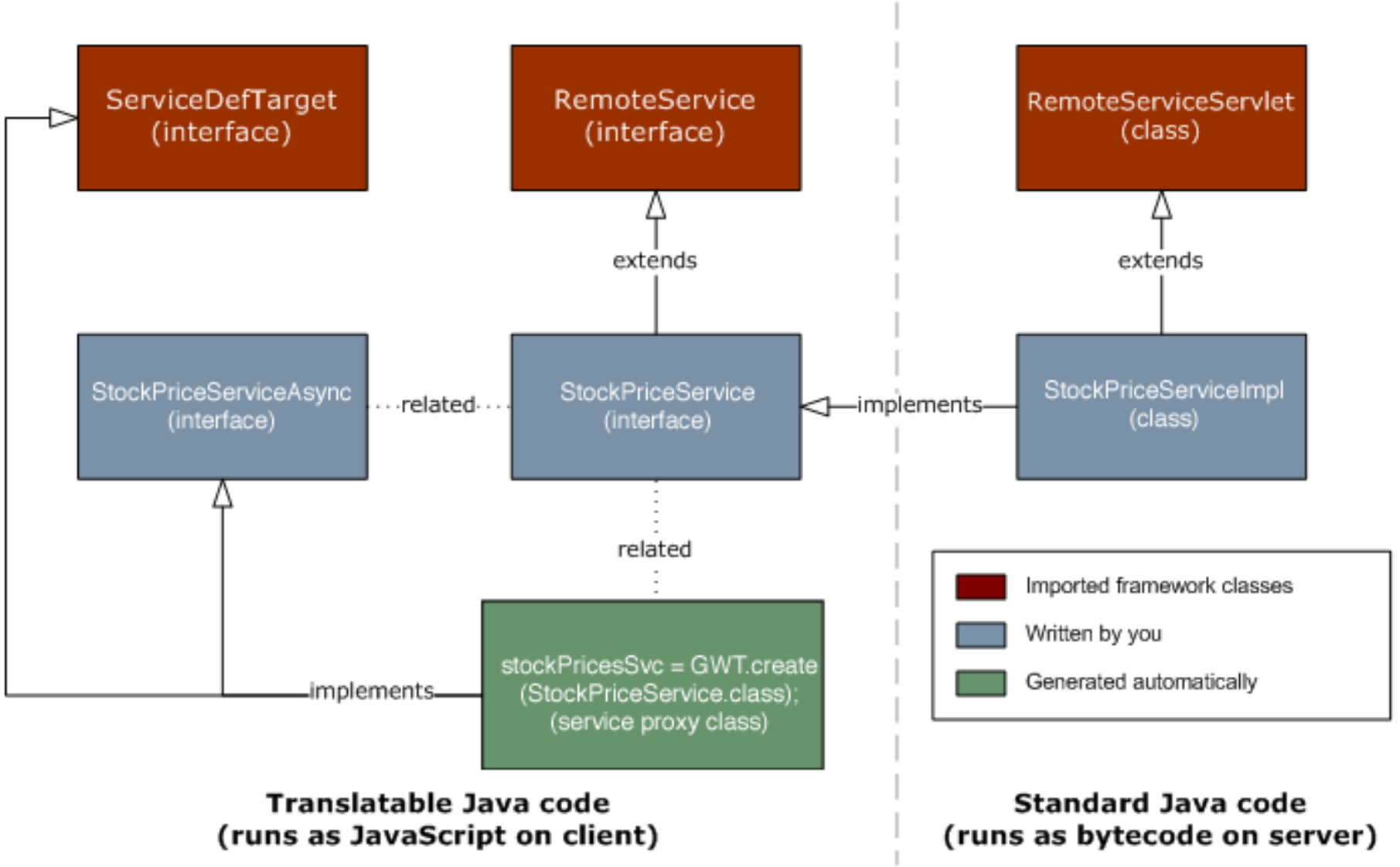


Příklad GWT – strom

```
// Add a handler that automatically generates some children
dynamicTree.addOpenHandler(new OpenHandler<TreeItem>() {
  public void onOpen(OpenEvent<TreeItem> event) {
    TreeItem item = event.getTarget();
    if (item.getChildCount() == 1) {
      // Close the item immediately
      item.setState(false, false);
      // Add a random number of children to the item
      String itemText = item.getText();
      int numChildren = Random.nextInt(5) + 2;
      for (int i = 0; i < numChildren; i++) {
        TreeItem child = item.addItem(itemText + "." + i);
        child.addItem("");
      }
      // Remove the temporary item when we finish loading
      item.getChild(0).remove();
      // Reopen the item
      item.setState(true, false);
    }
```



Asynchronní volání na server



https://developers.google.com/web-toolkit/doc/latest/tutorial/RPC?hl=cs

Asynchronní volání na server - kód

```
public class StockPriceServiceImpl extends RemoteServiceServlet
implements StockPriceService {
  public StockPrice getPrices(..) {
    ...
    return prices;
  }
}
```

```
private void refreshWatchList() {
   stockPriceSvc =
      GWT.create(StockPriceService.class);

  // Set up the callback object.
  AsyncCallback<StockPrice[]> callback =
      new AsyncCallback<StockPrice>() {
      public void onFailure(Throwable caught) {
        // TODO: Do something with errors.
      }

      public void onSuccess(StockPrice[] result) {
        updateTable(result);
      }
    };

   // Make the call to the stock price service.
   stockPriceSvc.getPrices(..., callback);
  }
```

Příklad GWT – internacionalizace

ErrorMessages.properties

```
permissionDenied = User ''{0}'' has security clearance ''{1}'' and cannot access ''{2}''
```

ErrorMessages.java (generated)

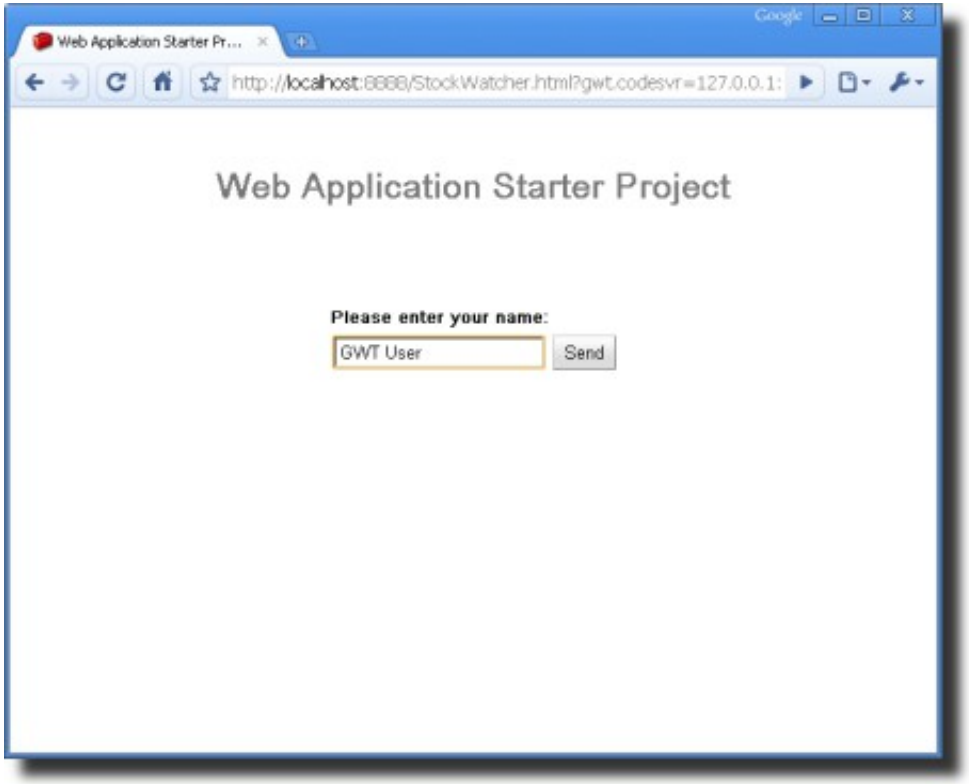
```
public interface ErrorMessages extends Messages {
  String permissionDenied(String username, String securityClearance,
      String inaccessibleResource);
}
```

Example.java

```
private ErrorMessages errorMessages = GWT.create(ErrorMessages.class);
formattedMessage.setText(errorMessages.permissionDenied(user, clearance, resource));
```

Jak začít s GWT

- **webAppCreator -out StockWatcher -junit "C:\eclipse\plugins\org.junit_3.8.2.v200706111738\junit.jar"**
  com.google.gwt.sample.stockwatcher.StockWatcher
  - vygeneruje projekt
- **ant devmode**
  - Running the development mode code server

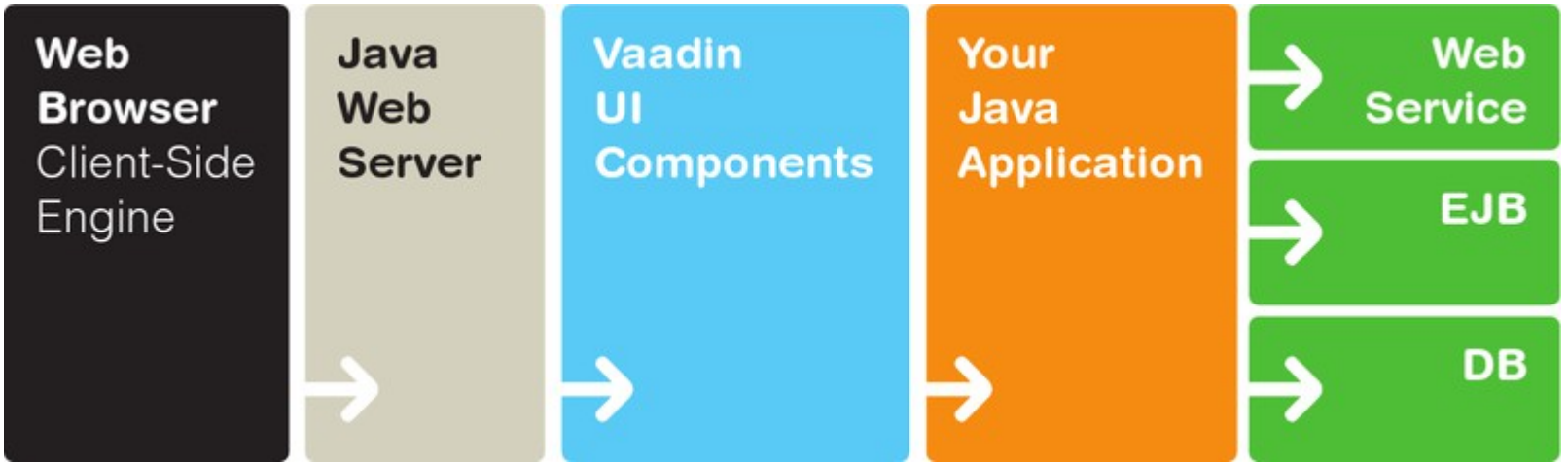


Linky

- http://code.google.com/intl/cs/webtoolkit/
  - home
- http://gwt.google.com/samples/Showcase/
  - sada funkčních příkladů se zdrojáky

Vaadin

- Vaadin zjednodušuje a urychluje práci s GWT, takže získává podobné vlastnosti jako JSF; zavádí i vlastní komponenty

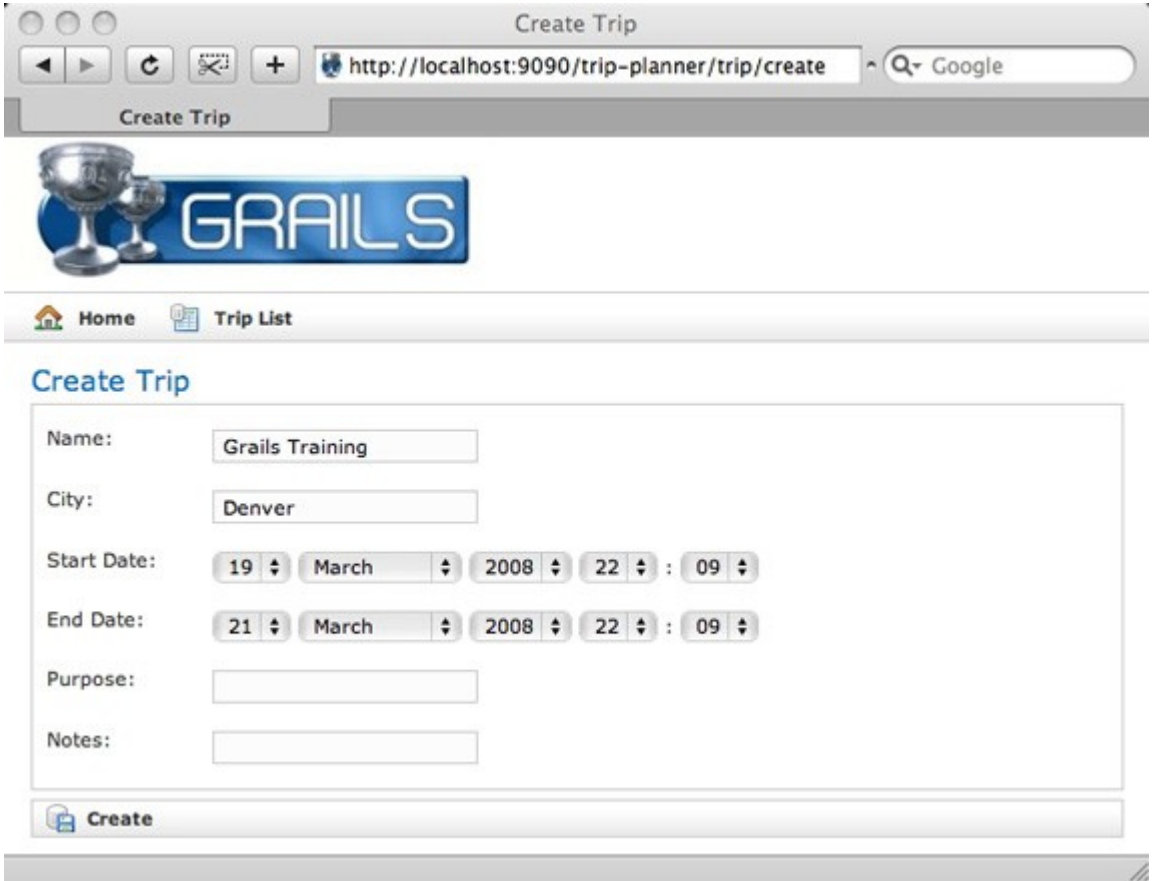


Vaadin příklad

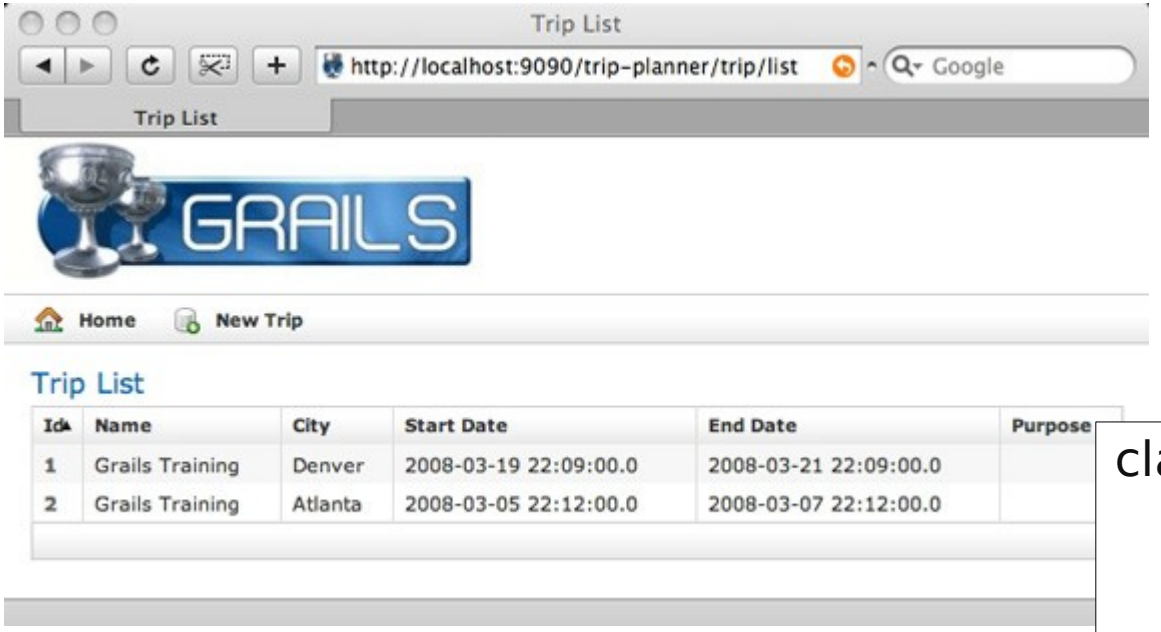
```
public class HelloWorld extends com.vaadin.Application {
    /**
     * Init is invoked on application load (when a user accesses the application
     * for the first time).
     */
    @Override
    public void init() {
        // Main window is the primary browser window
        final Window main = new Window("Hello window");
        setMainWindow(main);
        // "Hello world" text is added to window as a Label component
        main.addComponent(new Label("Hello World!"));
    }
}
```

Grails (RoR)

- rychlý vývoj (rapid development)
- běží nad JVM, Groovy
- automaticky generuje CRUD stránky



Grails List




```
class Trip {
  String name
  String city
  Date startDate
  Date endDate
  String purpose
  String notes
}
```

```
class TripController {
    ...
    def list = {
        if(!params.max) params.max = 10
        [ tripList: Trip.list( params ) ]
    }
    ...
}
```

```
<g:each in="${tripList}" status="i" var="tr
  <tr class="${(i % 2) == 0 ? 'odd' : 'even'}" >
    <td>
      <g:link action="show" id="${trip.id}">${trip.id?.encodeAsHTML()}</g:link>
    </td>
  </tr>
</g:each>
```

Transforming IT Management

Apache Wicket

- slavný framework, ceněný pro svou jednoduchost a rychlost, lze dělat vlastní komponenty
- důraz na server-side stav

```
<html>
<body>
   <span wicket:id="message">Message goes here</span>
</body>
</html>
```

```
public class HelloWorld extends WebPage {
    public HelloWorld() {
        add(new Label("message", "Hello World!"));
    }
}
```

Drobnější témata

- Local vs. Remote
- Vícevrstvá architektura
- Global JNDI Names
- Calendar Based Time Services
- Embeddable Container
- Profily
- End-to-end

Local vs Remote

10!	L (ms)	R (ms)
10x	32	13
100x	17	57
1 000x	236	400
10 000x	566	990
100 000x	2,693	4,259
1 000 000x	24,498	42,594

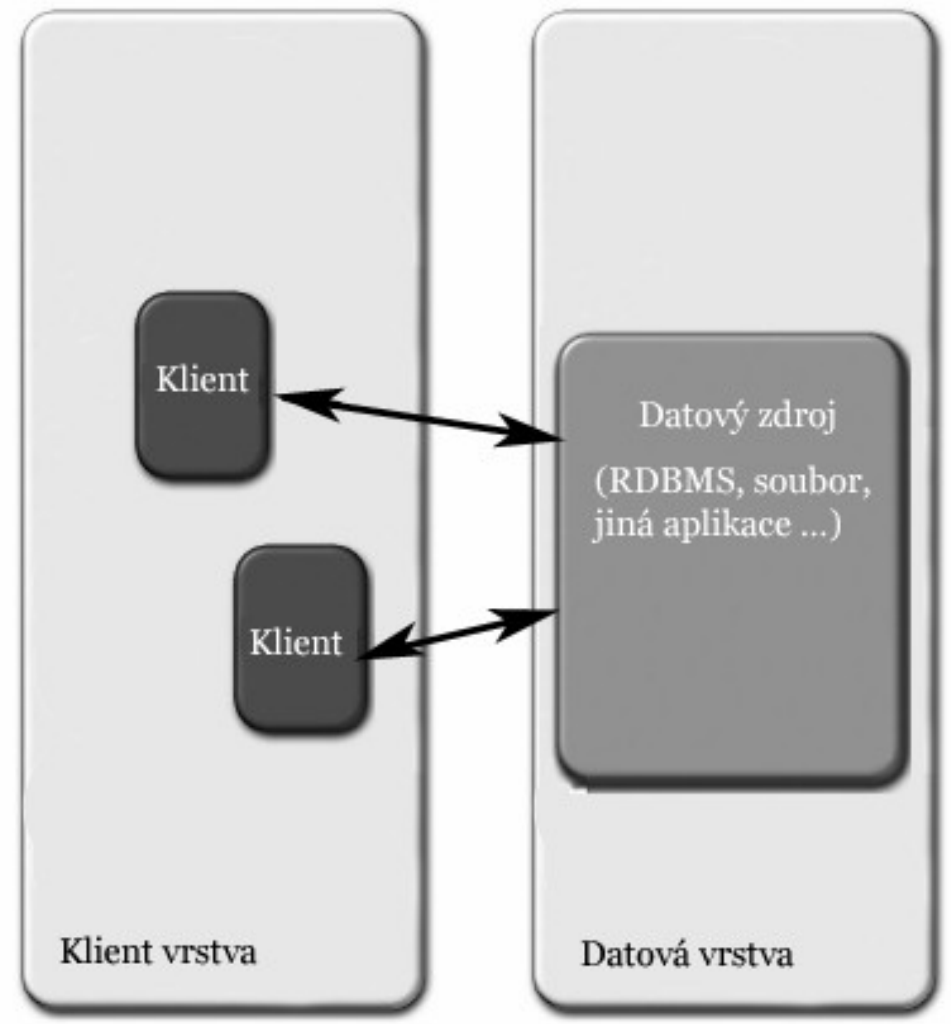
1 000!	L (ms)	R (ms)
10x	3	9
100x	41	29
1 000x	432	868
10 000x	4,326	6,216
100 000x	44,342	42,498
1 000 000x	401,302	418,383

100!	L (ms)	R (ms)
10x	0	1
100x	4	5
1 000x	49	124
10 000x	1,132	758
100 000x	6,543	8,942
1 000 000x	67,017	84,830

10 000!	L (ms)	R (ms)
10x	95	106
100x	565	307
1 000x	3,552	3,874
10 000x	38,518	38,518
100 000x	380,394	387,810
1 000 000x	4,689,103	4,768,913

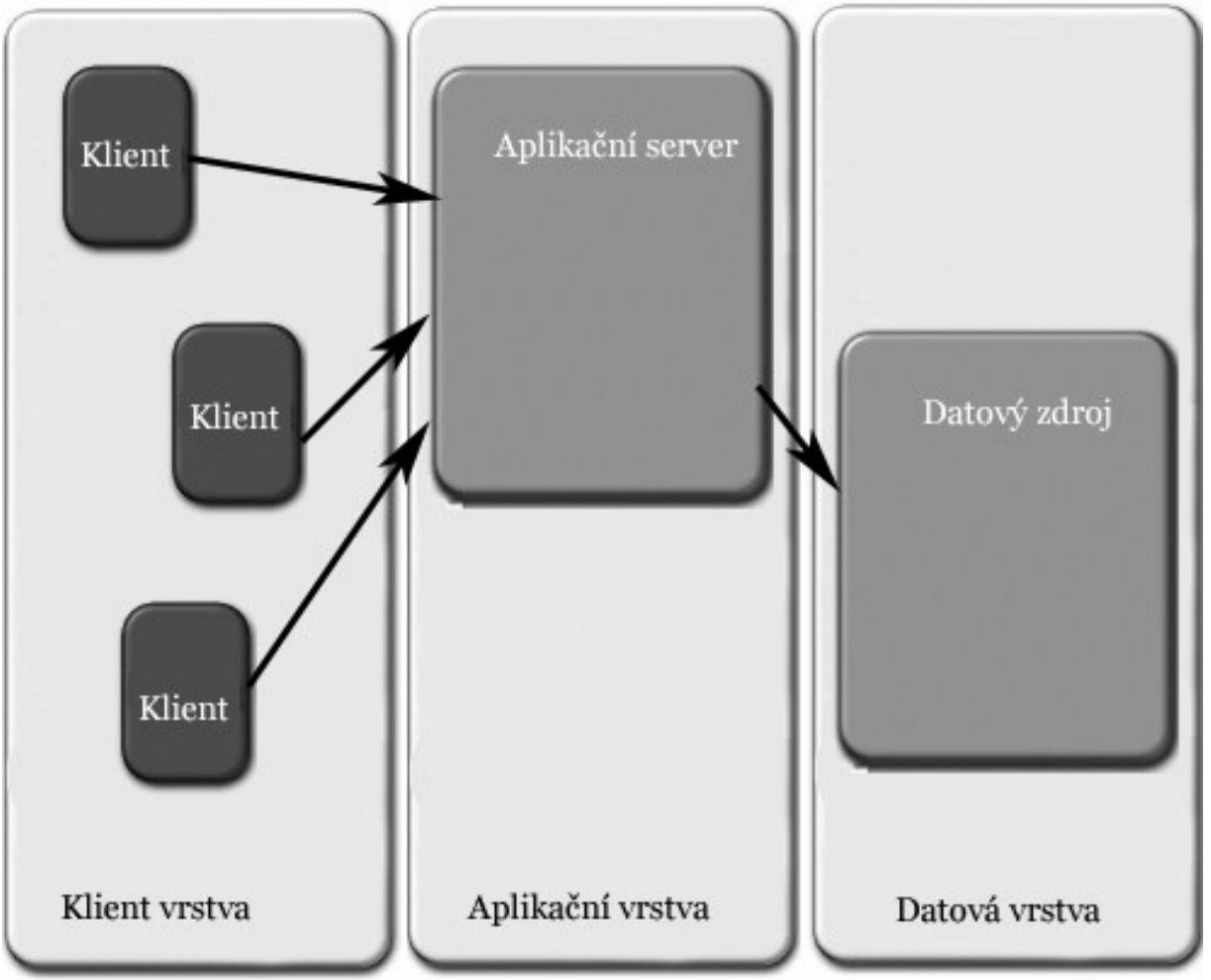
Třívrstvá architektura – 2 vrstvy

- obrázky jsem si půjčil od Dagiho (dagblog)
- 2 vrsvy byly prima v 90. letech
- kde to nestačí?



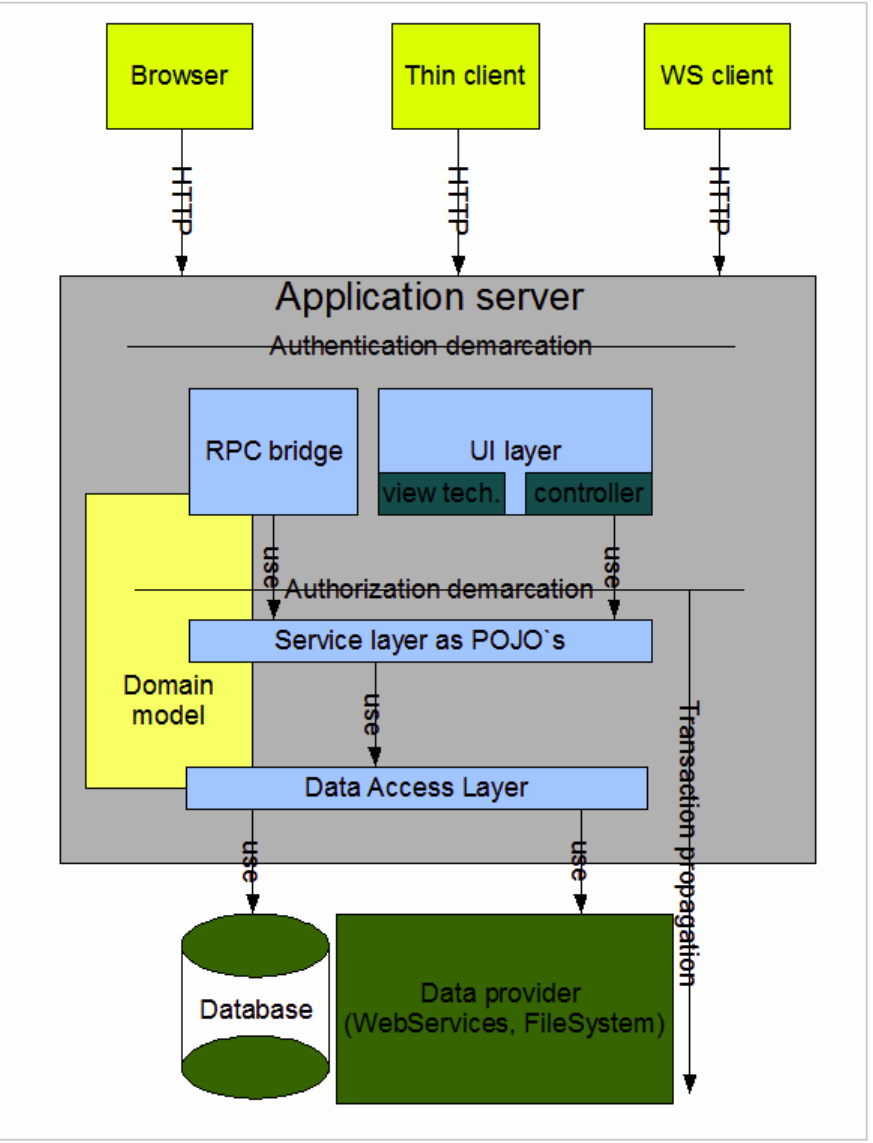
Třívrstvá architektura – 3 vrstvy

- zjednodušený model



Třívrstvá architektura – 3 vrstvy

- dnešní architektura



Global JNDI Names

- v současnosti jsou JNDI názvy specifické pro servery, takže je testování a deployment poněkud magie
  - Glassfish: com.os.ent.CustomerManagerRemote
  - JBoss: ejb3sampleapp/CustomerManager/remote
  - Oracle: CustomerManager
- ve 3.1 bude univerzání (alternativní) pojmenování:
  - java:global[/<application-name>]/<module-name>/<bean-name>#<interface-name>

Calendar Based Timer Services

- možnost spouštět akce v určitý čas, např. každé pondělí v 6:15
- podobně jako cron, diskuze se vede, jestli podporovat syntaxi cron (protože ji každý zná)

```
@Stateless
public class NewsLetterBean {
   @Schedule(minute=”15”, hour=”6”, dayOfWeek=”Mon”)
   public void sendNewsLetter() {
      ...
   }
}
```

EJB 3.1 Embeddable Container

- Problém s deploymentem lze řešit pomocí tzv. embeddable containeru. Neběží jako server, je iniciován kódem. Must have pro testování:

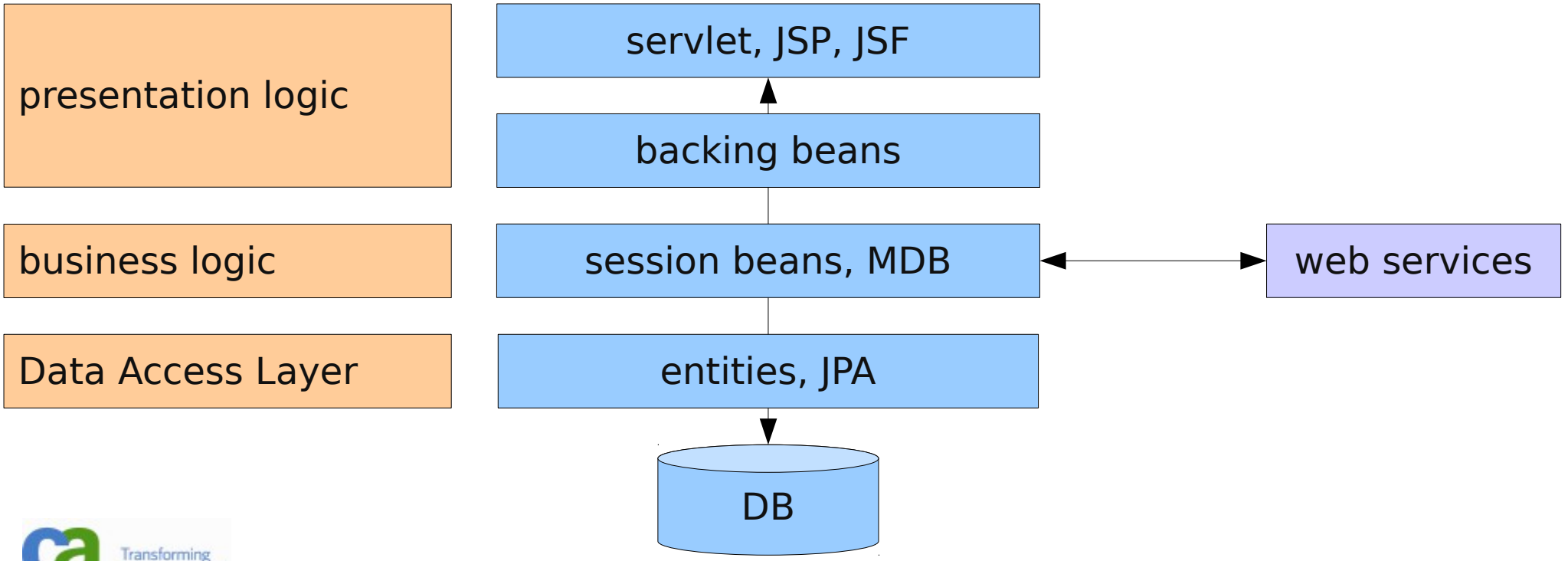
```
@Test
    public void testStates() throws Exception {
        Properties props = new Properties();
        props.setProperty(Context.INITIAL_CONTEXT_FACTORY,
                "org.apache.openejb.client.LocalInitialContextFactory");

        InitialContext context = new InitialContext(props);

        StateRegistry stateOne = (StateRegistry) context
                .lookup("StateRegistryBeanLocal");

        StateRegistry stateTwo = (StateRegistry) context
                .lookup("StateRegistryBeanLocal");

        stateOne.setState("MaryLand", "MD");
        stateTwo.setState("Virginia", "VA");
```

Profily JEE

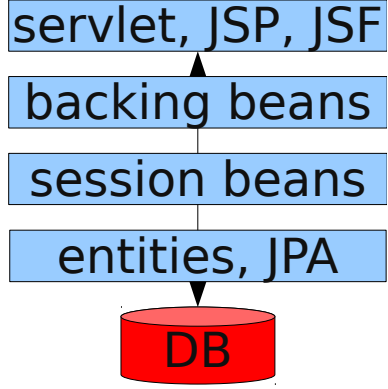
- Již nyní je JEE opravdu široká technologie s mnoha požadavky, které většina aplikací ani nevyužije (viz oblíbený terč CORBA)
- Pro JEE 6 je navržena myšlenka profilů, kdy kontejner musí implementovat určité technologie; není potřeba implementovat úplně všechny.
- Aktuálně je definován pouze Web Profile,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

End-to-end

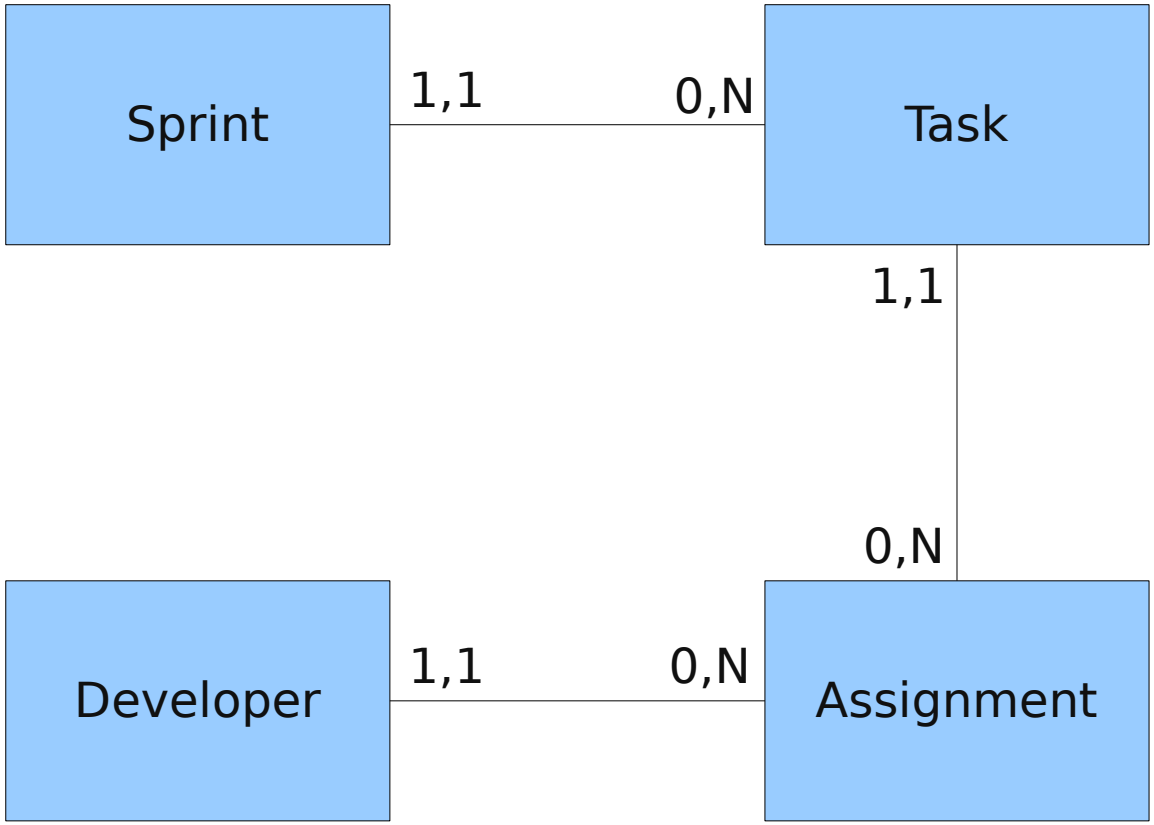
- Postup od začátku do konce... všechno



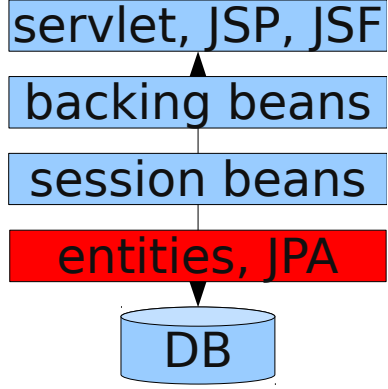
Databáze



- ER

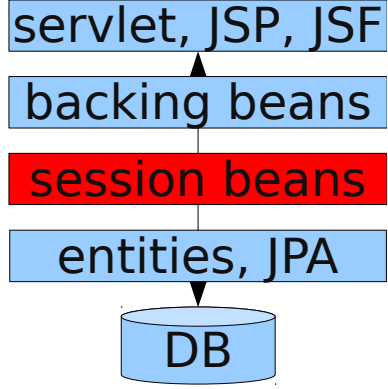


JPA (DAO, DAL)



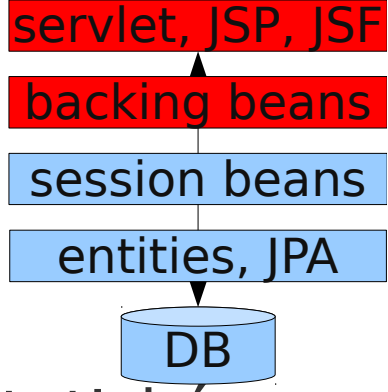
- prostě vygenerujeme
- doplníme @GeneratedValue(strategy=GenerationType.IDENTITY) a případně opravíme
- doporučuji mít dopředu připravená data (snáze se kontroluje)

Session beans (business logic)



- budeme potřebovat přístup k Entity manageru:
  @PersistenceContext
  private EntityManager em;
- samozřejmě necháme IDE generovat tyto části
- funčnost ověříme pomocí web services (buď z webového interface Glassfishe nebo necháme NB vygenerovat ws clienta)

Mock prezentační vrstvy



- vyrobíme webové stránky, jak mají vypadat, se statickým obsahem.
- Postupně je budeme nahrazovat funkčním obsahem a psát backing beany.
- Nepište velké části bez dalších návazností, předejdete tak velkým přepisům.
- Pište vždy funkcionalitu vertikálně, od session beans, backing bean až po JSF.

A rozšiřujeme...

- Budeme dále rozšiřovat; když už máme základ aplikace, přidávání dalších stránek je radost!
- Postupujte inkrementálně, práce roste viditelně pod rukama a motivuje vás k dalšímu vývoji! Rozdělená práce, kde dlouho nevidíte výsledek, demotivuje!
- Nesnažte se mít úchvatnou aplikaci hned z počátku, strávíte příliš času nad nedůležitými částmi a nakonec nebude aplikace hotová.
- Soustřeďte se na předem dohodnutou funkcionalitu. Až ji budete mít hotovou, můžete ji zkusit vylepšit, případně dodat další.

Linky

- 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