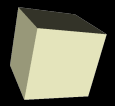
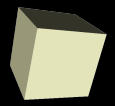
SCC-263
Técnicas de Programação para WEB

Rodrigo Fernandes de Mello
http://www.icmc.usp.br/~mello
mello@icmc.usp.br



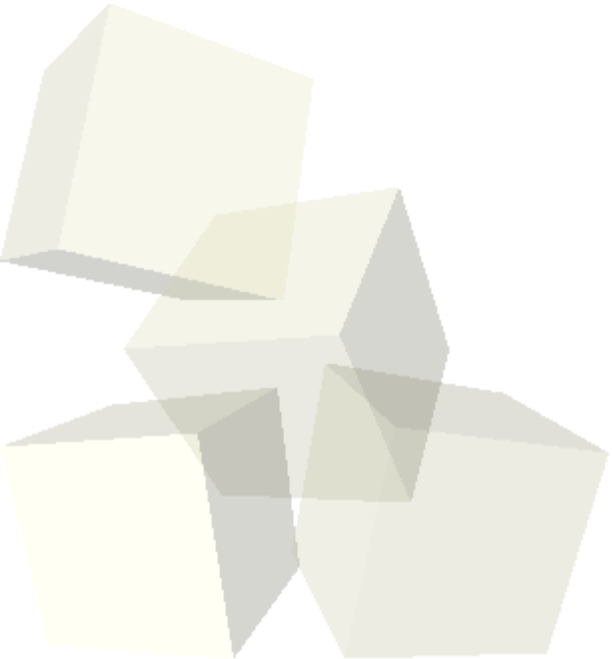
Cronograma

- Fundamentos sobre servidores e clientes
- Linguagens Server e Client-side
- HTML
- CSS
- Javascript
- JSP
- JavaBeans
- Ant
- Servlets
- Hibernate
- Maven
- XML, SAX e DOM
- Axis e Web Services
- JSF
- EJB



Referências

- http://java.sun.com
- http://www.apache.org
- Material e Livro: http://www.icmc.usp.br/~mello/downloads.php
- Mukhi, Vijay; Java servlets JSP, Makron Books, 2002

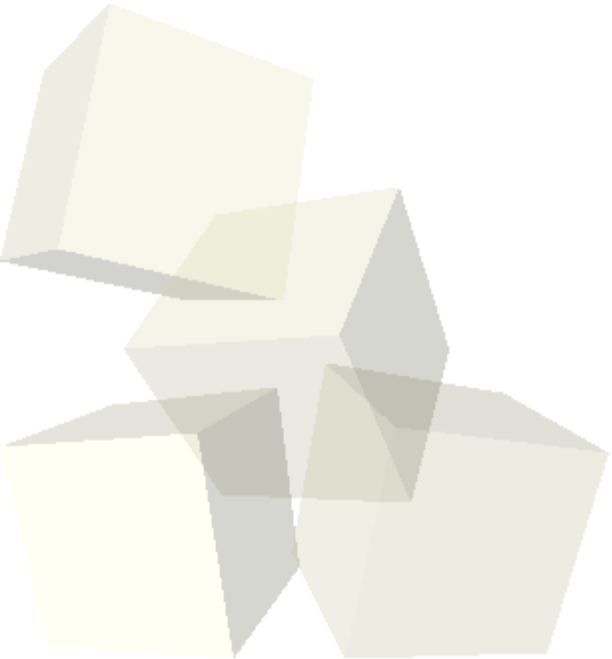


Critérios de Avaliação

- Média = 3/(1/P1 + 1/P2 + 1/MT)
- Não haverá SUB
- Trabalhos – MT é uma média simples de trabalhos

Datas de Provas

- P1 – 15/04/2011 – Sexta-feira
- P2 – 20/06/2011 – Segunda-feira



Atendimento

- Sala 3-162
  - Terças das 17:30 hs – 20:30 hs
- Atendimento Estagiário PAE
  - ???

Cliente e Servidor

Cliente
•Envia requisições para o servidor e recebe resultados
•Algumas linguagens/frameworks client-side tais como:
- ActiveX
- JavaScript
- AJAX

Servidor
•Responde requisições de usuários
•Executa linguagens server-side tais como:
- Perl
- Phyton
- PHP
- JSP/Servlets
- etc

Servidores WEB

- Há vários servidores para aplicações WEB, o mais comum é o Apache (www.apache.org)
- Iremos utilizar uma variação do Apache, o Apache Tomcat que suporta, de maneira nativa, a execução de aplicações Java
- Se quisermos utilizar o Apache tradicional, basta habilitarmos um módulo de suporte a Java
  - O mesmo ocorre com outras linguagens tais como PHP, Python, Perl etc...
- Instalando e Configurando o Apache...

HTML e CSS?

- HTML (HyperText Markup Language) é uma linguagem para gerar conteúdo para navegadores WEB
  - O navegador do cliente “monta” esse conteúdo conforme enviado pelo servidor
- CSS (Cascading Style Sheets) é linguagem de script que auxilia na criação de layouts para páginas WEB
  - Permite a definição de fontes, tamanhos, cores etc
  - O navegador cliente recebe tais informações e apresenta conforme definido pelo servidor
- Resumo
  - Tais linguagens auxiliam na montagem de páginas WEB para apresentação por parte dos navegadores
  - Exemplos (html* e htmlcss*)

PHP, Python, Java etc...

- Essas linguagens podem ser utilizadas em conjunto com HTML e CSS para adicionar aspecto dinâmico aos sites tais como:
  - Acesso a bancos de dados
  - Execução de laços
  - Geração de relatórios etc
- Como era antes? (CGI – Common Gateway Interface)
- Iremos abordar a disciplina utilizando Java o que permite aplicação de dois frameworks:
  - JSP – Java Server Pages
  - Servlets

O que é Servlet?

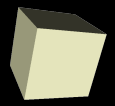
- A plataforma Java provê um framework para a construção de aplicações WEB utilizando superclasses
- Código é executado do lado do servidor e resultados retornados para o navegador cliente
- Exemplos (servlets*)
- Trabalho 1

O que é JSP?

- Instruções Java podem ser embutidas em código HTML
- Código é executado do lado do servidor e resultados retornados para o navegador cliente
- A primeira vez que uma página JSP é chamada, ela é compilada e transformada em um Servlet, por isso demora para retornar resultados na primeira execução...
- Exemplos (jsp*)

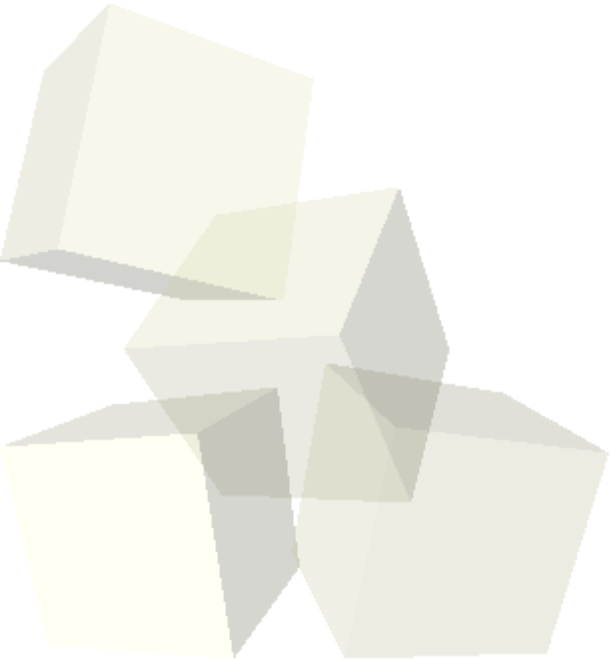
O que são JavaBeans?

- Classes (também denominados componentes) que auxiliam no desenvolvimento de aplicações WEB
- Características:
  - Devem ter um construtor vazio
  - Permitem criar objetos e chamar métodos de forma implícita
  - Objetos encapsulam código
  - Alteração de código é feita nas classes e propagada em todo o sistema
    - → Não há a necessidade de alterar inúmeras páginas
- Exemplos (jspjb*)



Bancos de dados

- MySQL
- PostgreSQL
- Oracle etc...
- Criação de bancos de dados em PostgreSQL



JavaBeans: Persistência de BD

- JavaBeans podem encapsular todo o código que faz acesso a bancos de dados
  - Design Pattern
- Isso simplifica o desenvolvimento de páginas WEB
- Exemplos:
  - Criando conexões fora da classe (utilizando interface)
  - Criando conexões com Design Pattern Factory (e utilizando interface)
  - Jspdb*
- Trabalho 2

Modelo MVC (Model-View-Controller)

- MVC é um padrão usado em engenharia de software:
  - Isola a lógica de negócios da interface de usuário
  - Aplicação fica mais fácil de ser modificada:
    - Tanto no aspecto visual quanto em termos de regras de negócio
    - O Model representa os dados da aplicação e as regras de negócio utilizadas para trabalhar sobre tais dados. Ex: JavaBeans
    - A View corresponde aos elementos da interface de usuário tais como textos, caixas de entrada de dados etc. Ex: JSP e HTML
    - O Controller gerencia detalhes de mensagens oriundas de ações de usuários. Ex: Servlets
- Exemplos (mvc*)
- Trabalho 3

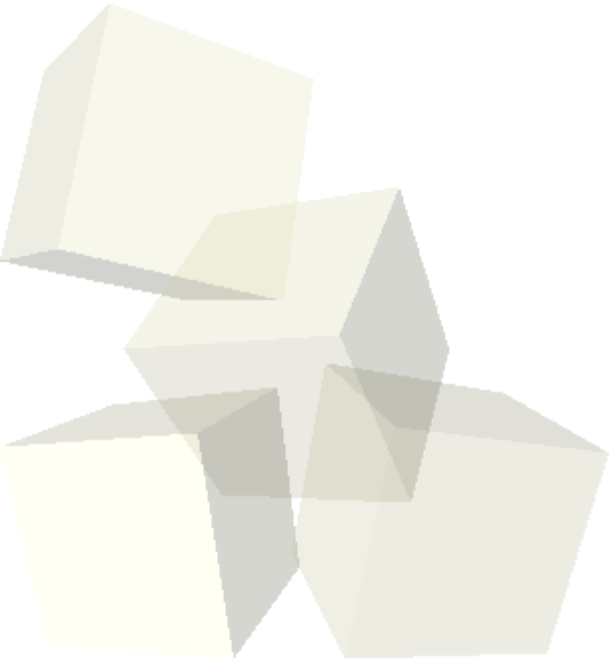
Apache Ant

- Apache Ant é uma ferramenta para compilação de aplicações Java
  - É um tipo de Make
  - Por que outro Make? Permite compilação multi-plataforma. Alguns comandos usando no Makefile tornam-o dependente até mesmo do Shell
  - Makefile necessitam de TABs e ocorrem erros...
  - Adota um arquivo XML para realizar compilação
- Instalar Apache Ant
- Exemplos:
  - Aplicação Desktop simples
  - Partindo para Web Application (WAR) – Automatizando Deployment no Apache Tomcat
  - ant*

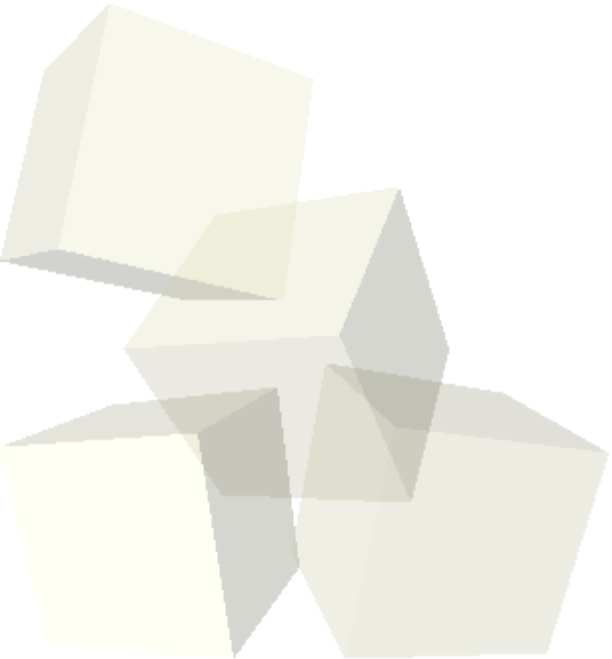
- Um exemplo com Log4j
  - Código pode conter debug
  - Configura-se comportamento do debug via arquivo log4j.properties
    - Ligar e desligar
    - Evita problemas de desempenho em ambiente de produção, contudo permite debug, caso necessário
  - Permite hierarquia de logs
    - Pode-se definir para qual nível será feito output
  - Target
    - Arquivo, OutputStream, java.io.Writer, um servidor de log4j remoto, um daemon syslog etc

Hibernate: Framework de Persistência

- Framework que oferece persistência de Objetos em bancos de dados
- Reduz custo de desenvolvimento
- Exemplos...
- Trabalho 4

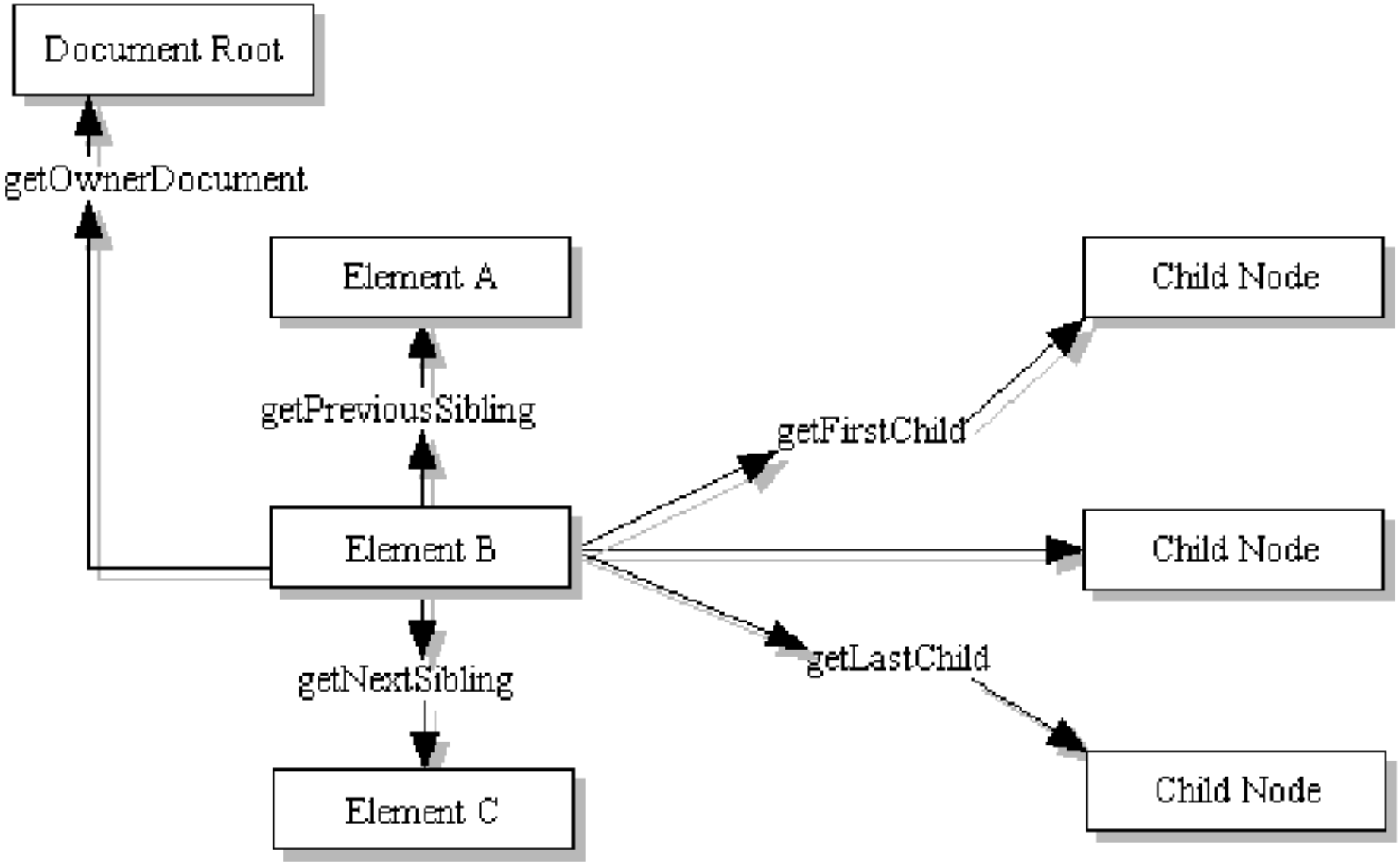


- JavaServer Faces (JSF) é um padrão para construção de aplicações Web
- JSF simplifica o desenvolvimento do View e aspectos de Controller do modelo MVC
- Exemplos...
- Trabalho 5



XML, SAX e DOM

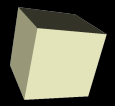
- XML (eXtensible Markup Language)
  - Simplificou a troca de informações entre aplicações
  - Como era antes? Ex: Recebimento de boletos bancários
    - Adição de novos campos???
    - Remoção de campos???
    - Ordem dos campos???
  - ???
- DTD
  - ???
- Como ler o conteúdo de um XML?
  - Parsers:
    - SAX (Simple API for XML)
    - DOM (Document Object Model)
  - Quem fornece as APIs → Sun Microsystems, Apache, etc
- Exemplos: sax* e dom*


Document Root
getOwnerDocument
Element A
getPreviousSibling
Element B
getNextSibling
Element C
getFirstChild
Child Node
Child Node
getLastChild
Child Node

Axis e Web Services

- O Apache Axis é um framework de código aberto, baseado na linguagem Java e no padrão XML, utilizado para construção de web services no padrão SOAP
- Através do Axis os desenvolvedores podem criar aplicações distribuídas. Além da versão para Java, existe uma implementação baseada na linguagem C++
- Gera automaticamente o arquivo WSDL (Web Service Description Language). O WSDL contém a definição da interface dos web services.

- SOAP
  - SOAP (originado do acrônimo inglês Simple Object Access Protocol) é um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída, utilizando tecnologias baseadas em XML
  - Sua especificação define um framework que provê maneiras para se construir mensagens que podem trafegar através de diversos protocolos de comunicação (por exemplo: HTTP/TCP/IP) e que foi especificado de forma a ser independente de qualquer modelo de programação ou outra implementação específica
  - Servidores SOAP geralmente são implementados sobre servidores HTTP



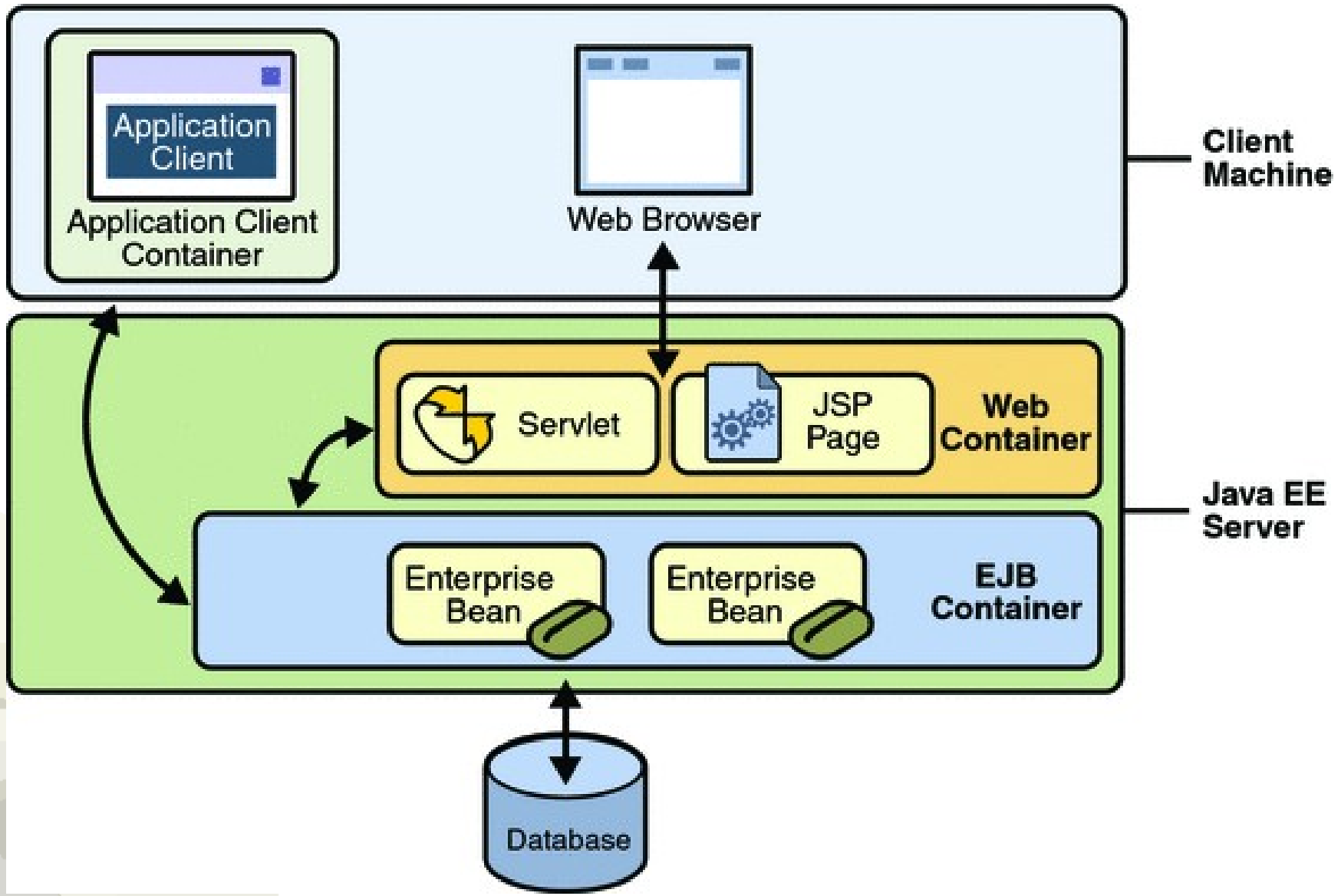
Axis e Web Services

- SOAP
  - As mensagens SOAP são documentos XML que aderem a uma especificação fornecida pelo órgão W3C
  - O primeiro esforço do desenvolvimento do SOAP foi implementar RPCs sobre XML
- Instalando Apache Axis
- Exemplos: ws*
- Trabalho 6

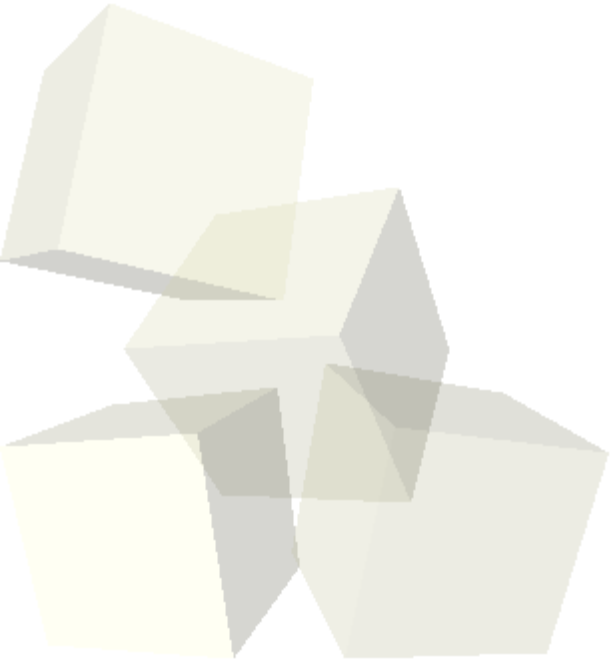
Enterprise Java Beans 3.0

- Arquitetura para a construção de aplicações utilizando componentes modulares
  - Server-side
  - Especificação originalmente desenvolvida em 1997
  - Especificações:
    - EJB 1.0
    - EJB 1.1
    - EJB 2.0
    - EJB 2.1
    - EJB 3.0
- Visa, basicamente, oferecer um Container que simplifica o desenvolvimento e suporta:
  - Transações, persistência, controle do concorrência, criptografia, autenticação, chamadas remotas, serviço de mensagens etc

Enterprise Java Beans 3.0



- JNDI
  - Suporte Java Naming and Directory Services
  - Encontrar objetos registrados

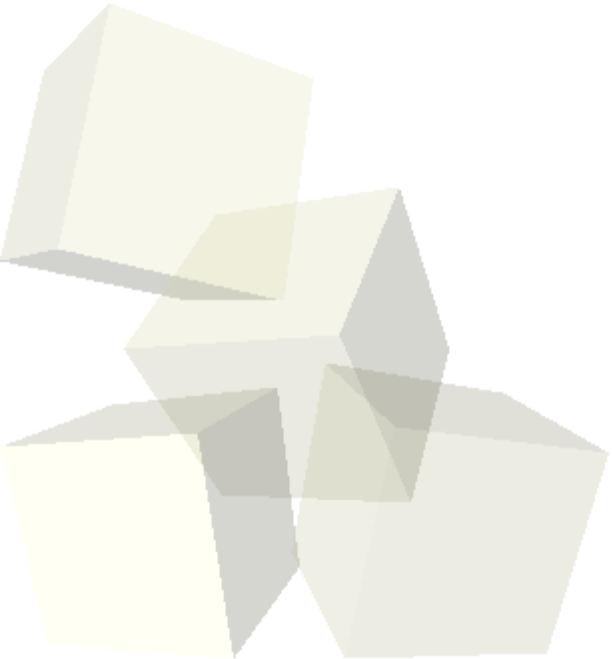


- Tipos de componentes:
  - Stateful Session Beans
    - Objetos distribuídos com estado
    - Fazem track do processo que os chama durante execução
    - Somente um cliente tem acesso ao Bean
    - Utilizado quando há um processo de múltiplos passos
      - Ex: um carrinho de compras
  - Stateless Session Beans
    - Objetos distribuídos sem estado
    - Permitem acesso concorrente ao Bean
    - O conteúdo de variáveis não é mantido entre chamadas
      - Isso os deixa mais simples e mais leves que os Stateful
    - Utilizado quando há um processo de um passo
      - Ex: enviar um email
  - Entity Beans
    - Acesso a banco de dados
    - Persistência gerenciada pelo Container ou manual

Maven: Evolução do Ant

- Não há a necessidade de termos toda uma infraestrutura instalada para desenvolver nossos softwares
- Automatiza download de pacotes da Internet de acordo com as versões esperadas para a compilação do software
- Faz compilação
- Gera pacote final
- Testa software
- Ou seja, o Maven gerencia o desenvolvimento de projetos
- Exemplos...
- Trabalho 7

- Download do Maven
- Instalação
- Alterar PATH para o diretório bin
- Criar projeto
  - mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=my-app



Maven: Estrutura de Diretórios

```
my-app
|-- pom.xml     (project object model)
`-- src
    |-- main        (código fonte)
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                   `-- App.java
    `-- test          (testes automatizados com jUnit)
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

Maven: Ciclo de Vida de Build

- validate: valida o projeto e informações necessárias
- compile: compila o código fonte do projeto
- test: testa o código fonte compilado usando o jUnit
- package: gera um pacote JAR, WAR e EAR do projeto
- integration-test: faz deployment do pacote no ambiente de testes
- verify: verifica se o pacote é válido segundo critérios
- install: instala o pacote install em repositório local (incluindo dependências)
- deploy: faz deployment para ambiente de produção
- site: gera site com documentação do projeto
- clean: limpa o build

- Mais comuns:
  - mvn compile
  - mvn test
  - mvn package
  - mvn install
    - → Repositório local
  - mvn site
- Dependências:
  - http://repo1.maven.org/maven2
  - groupId
  - artifactId
  - version
  - scope → test, compile e runtime