Globalcode®
The Developers Company

Open-source Education

Introdução à plataforma J2EE

Iniciativa Globalcode

Mini-cursos Globalcode

MC1 – Introdução à plataforma Java
MC2 – Sintaxe da linguagem e orientação a objetos com Java
MC3 – Introdução à plataforma J2EE – Java 2 Enterprise Edition
MC4 – Desenvolvimento de aplicativos Web com Java
MC5 – J2EE modelando arquiteturas para demandas de 10 a mais de 10.000 usuários
MC6 – Java e mainframe: analogias, integrações e arquiteturas
MC7 – Metodologias de desenvolvimento para Java e UML
MC8 – Desenvolvimento Web com design-patterns e Struts
MC9 – Desenvolvimento de componentes Enterprise JavaBeans
MC10 – Planejamento e execução de stress-test
MC11 ao MC13 – Preparatórios para certificações Java

Agenda – Parte Teórica

1. Introdução
2. Vantagens e casos de sucesso
3. J2EE Vs. Microsoft .NET
4. Composição técnica da arquitetura J2EE
5. Camada Web
6. Camada EJB (componentes de negócio)
7. Clustering
8. Modelos de arquiteturas
9. Demonstrações

A Globalcode

The Developers Company

Educação treinamentos gratuitos, vídeo-aulas, palestras em empresas e universidades, cursos individuais, carreiras e serviços de consultorias pontuais e mentoring;

Pesquisa desenvolvimento de experiências com publicações em conferências internacionais - eXPerience Group -, convênio com ITA e IPEN;

Produção de software pequena fábrica de desenvolvimento de componentes Java, em expansão para 2006;

Palestrante / Instrutor

Vinicius Senger – vinicius@globalcode.com.br

- Sócio e fundador da Globalcode, foi instrutor e consultor da Sun e Oracle no Brasil;
- Trabalhou em projetos de grande porte em bancos. Começou a programar com 8 anos e trabalha com desenvolvimento de softwares profissionalmente desde os 13 anos;
- Certificações: Sun Java Programmer / Sun Enterprise Architect P1, Microsoft Certified Professional, Microsoft Certfied Trainner;

Motivação

- Empresas e desenvolvedores de software procuram cada vez mais:
  - Desenvolvimento padronizado;
  - Mais de uma opção de fabricante de determinada tecnologia;
  - Reaproveitamento do legado;
  - Soluções com alta disponibilidade e confiabilidade através de cluster de servidores;
  - Conhecimentos e experiência para novos projetos;

Agenda – Parte Teórica

1. Introdução
2. Vantagens e casos de sucesso
3. J2EE Vs. Microsoft .NET
4. Composição técnica da arquitetura J2EE
5. Camada Web
6. Camada EJB (componentes de negócio)
7. Clustering
8. Modelos de arquiteturas
9. Demonstrações

Introdução

- J2EE é um padrão para middlewares;
- Um middleware está para o código assim como um database server está para o dado
- Um servidor J2EE é um hotel de código e componentes Java;
- J2EE representa uma plataforma flexível para o desenvolvimento de softwares em múltiplas camadas de pequeno, médio e grande porte;

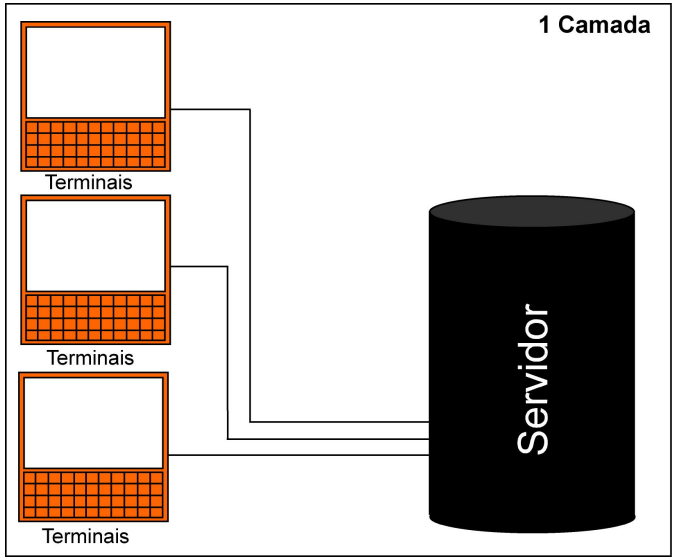
Introdução

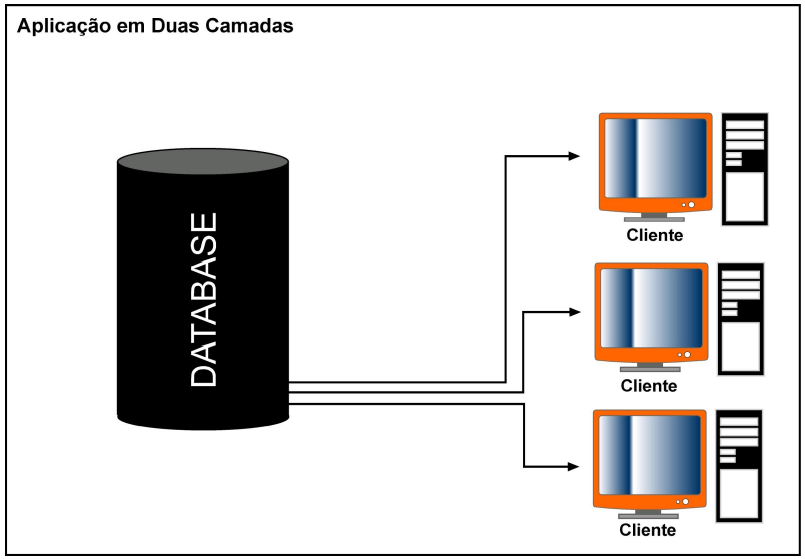
- J2EE: padrão de componentização, um hotel de objetos 5 estrelas;

- Ofertas de um container:
  - Escalabilidade, gestão de memória, ciclo de vida de objetos e estado de objetos;
  - Conexões, Transações, Serviço de nomes;
  - Segurança;
  - Prova de falhas para beans de entidade;
  - Integração e WebServices;
  - Agendamento (Session Timed Objects)
  - Clustering, alta disponibilidade e confiabilidade

Introdução

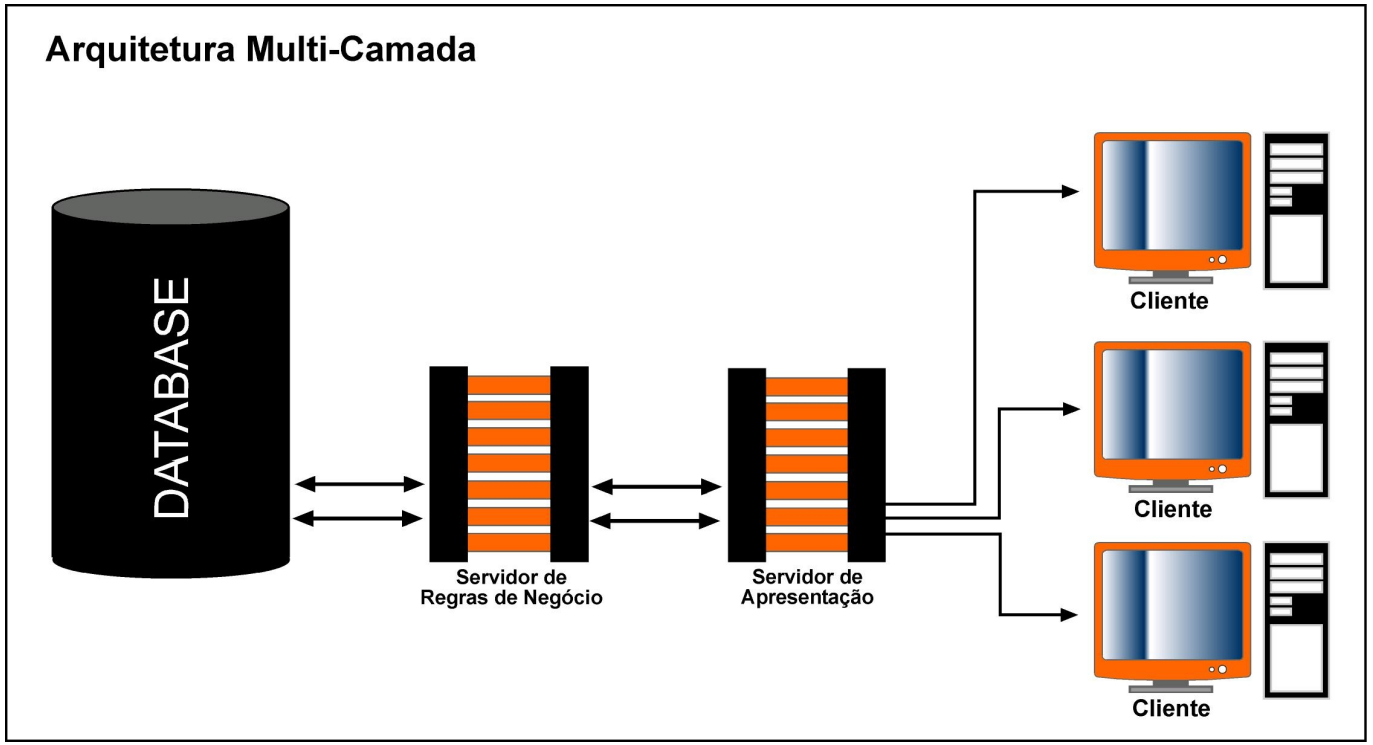
- Os softwares corporativos utilizam, arquiteturas monolíticas ,em duas camadas / client-server ou n-tier:





Introdução

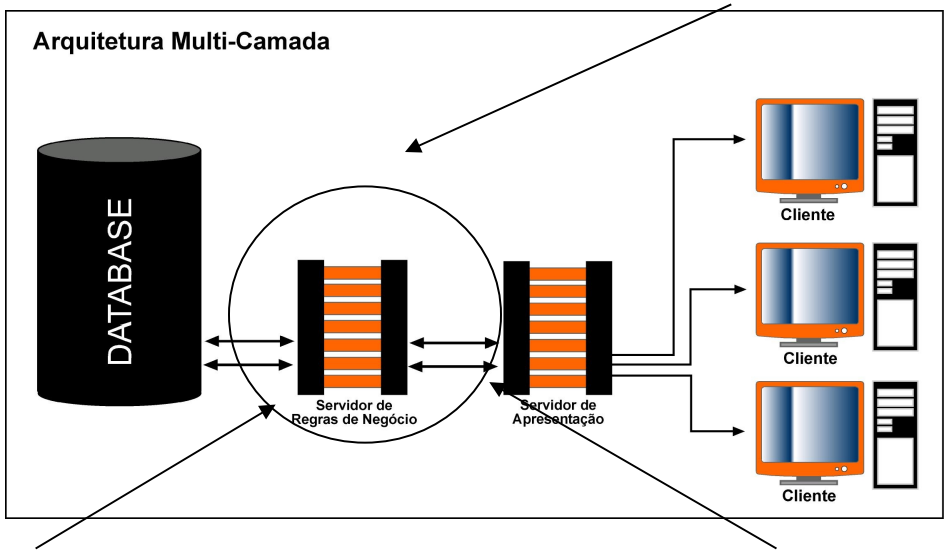
- J2EE é uma especificação para desenvolvimento em múltiplas camadas para soluções Web e não Web:



Players

Microsoft

Antigo COM, ActiveX. Atual: COM+ e DCOM no .NET e DNA



J2EE – www.jcp.org

Object Management Group

WebSphere, WebLogic, Macromedia JRun, JBoss, Jeronimo, Oracle OC4J, e mais 45 outras opções

Padrão CORBA

Agenda – Parte Teórica

1. Introdução
2. Vantagens e casos de sucesso
3. J2EE Vs. Microsoft .NET
4. Composição técnica da arquitetura J2EE
5. Camada Web
6. Camada EJB (componentes de negócio)
7. Clustering
8. Modelos de arquiteturas
9. Demonstrações

Vantagens e casos de uso

- Independente de plataforma e sistema operacional: mainframe, celular, palmtops, PC com Linux, Windows, Apple, Sun Solaris, IBM Aix, HP-UX, QNX, etc.

- Escalabilidade: nasceu na grande rede. EXISTEM projetos com mais de 20.000 usuários simultâneos, 5 milhões de transações dia entre outros;

- 19 implementações oficiais de servidores J2EE completos;

Vantagens e casos de uso

- Extensibilidade e reuso: através da orientação a objetos e design-patterns pode-se atingir um alto índice de reuso;

- Java em todos os lugares: desenvolvimento de front-ends, processamento de negócio, integração de plataformas, stored procedures e triggers, robots, etc.

- Grande número de API's: reconhecimento de voz, acessibilidade, XML, WebServices, transações, mensagens assíncronas, entre outros;

- Proteção de investimento;

Vantagens e casos de uso

Casos de uso:

- Imposto de renda
- Sistema de Pagamento Brasileiro
- Banco do Brasil
- DuPont
- Natura
- Banco JP Morgan
- Avon
- Caixa Econômica Federal
- eBay
- SUS

Agenda – Parte Teórica

1. Introdução
2. Vantagens e casos de sucesso
3. J2EE Vs. Microsoft .NET
4. Composição técnica da arquitetura J2EE
5. Camada Web
6. Camada EJB (componentes de negócio)
7. Clustering
8. Modelos de arquiteturas
9. Demonstrações

J2EE vs. Microsoft .NET

- J2EE você tem liberdade de escolha, isso é bom e ruim;
- ASPX e C# realmente são bons;
- O Visual Studio é um IDE maduro;
- J2EE você tem **efetivamente** opções pagas e gratuitas de servidores e IDE's;
- .NET fica mais complexo quando você necessita desenvolver componentes Enterprise DCOM;

J2EE vs. Microsoft .NET

- Ambos geram código para facilitar o desenvolvimento;
- Vários casos de sucesso nas duas plataformas;
- Competição esta sendo produtiva e ambas as tecnologias vão conviver;
- Desenvolvedores Java acumulam maior cultura em P.O.O. e design-patterns;
- J2EE é economicamente escalável;

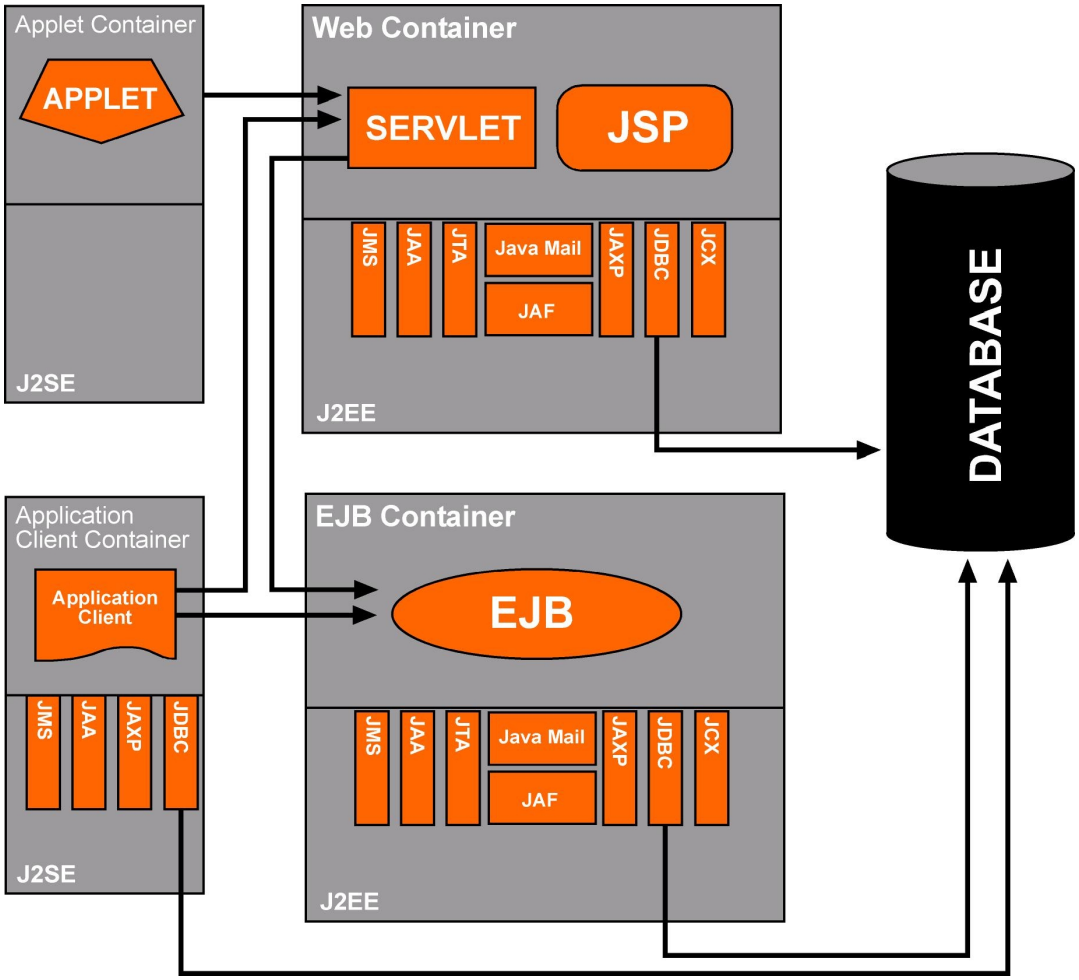
Agenda – Parte Teórica

1. Introdução
2. Vantagens e casos de sucesso
3. J2EE Vs. Microsoft .NET
4. Composição técnica da arquitetura J2EE
5. Camada Web
6. Camada EJB (componentes de negócio)
7. Clustering
8. Modelos de arquiteturas
9. Demonstrações

Java EE: composição

- J2EE é composto por dois diferentes servidores, tipicamente chamados de **container**;
- **Web Container**: servidor que vai trabalhar em conjunto com um HTTP Server para prover funcionalidades e páginas dinâmicas via HTTP;
- **EJB Container**: servidor de componentes transacionais Enterprise JavaBeans. Existem vários tipos de EJB: para representar processos, entidades, processos agendados e consumo de mensagens
- Mais de 30 opções de arquitetura!

J2EE: composição



J2EE: composição

- Lista de recursos e API's:

1. Páginas dinâmicas e controle Web: Java Servlets;
2. Páginas dinâmicas: Java Server Pages;
3. Reuso de scriptlets: Tag Library / Custom Tags;
4. Filtros Web: Servlet Filter;
5. Transação: Java Transaction Service
6. Banco de dados: JDBC 3.0 com pooling e recursos avançados de RDBMS;
7. Integração: Java Connector;
8. JavaMail;
9. JMX - Gerencimento;
10. EJB Session Bean: processos e código de negócio
11. EJB Entity Bean: entitdades persistentes;
12. EJB MessageDriven Bean: consumo de filas de mensagens;

Agenda – Parte Teórica

1. Introdução
2. Vantagens e casos de sucesso
3. J2EE Vs. Microsoft .NET
4. Composição técnica da arquitetura J2EE
5. Camada Web
6. Camada EJB (componentes de negócio)
7. Clustering
8. Modelos de arquiteturas
9. Demonstrações

Camada Web

- J2EE é sem dúvida a plataforma mais robusta para o desenvolvimento de aplicativos Web;
- Devemos utilizar um servidor Web (http server) e um Web Container (servidor de componentes Java);
- O servidor Web Java mais famoso é o Tomcat;
- O Tomcat é uma implementação de referência: tudo que funciona nele, funciona em outras implementações;

Camada Web

- Podemos desenvolver os seguintes componentes específicos para Web:

  - Java Servlet: é uma classe que é acionada por HTTP
  - JavaServer Page: é uma página HTML com código Java embutido;
  - JavaServer Faces: ambiente orientado por componentes e eventos de alta produtividade;
  - Tag Library: componentes Java utilizados através de tags em páginas HTML;
  - Filtros Web: componentes acionado para interceptar e analisar requisições Web;

Agenda – Parte Teórica

1. Introdução
2. Vantagens e casos de sucesso
3. J2EE Vs. Microsoft .NET
4. Composição técnica da arquitetura J2EE
5. Camada Web
6. Camada EJB (componentes de negócio)
7. Clustering
8. Modelos de arquiteturas
9. Demonstrações

Camada EJB

- EJB é um componente Java que pode ser acessado remotamente:

```
ContaCorrente conta =
ServidorEJBs.criar("ContaCorrente")
```

- EJB contém regras de negócio e processamento de dados;
- Tipicamente um EJB acessa banco de dados;
- Um EJB pode acessar mainframe por TCP/IP ou através de Connectors;

Camada EJB

- EJB tem contexto transacional e cuida do processo de commit / rollback;
- O uso de EJB's não é obrigatório;
- Quando usar EJB?
  - Alta escalabilidade;
  - Integração e compartilhamento de componente;
  - Uso de rich-clients sem Web Container;
  - Clientes não-Java
  - Flexibilidade na arquitetura

Camada EJB

- Para executar um EJB precisamos de um servidor / container de EJBs;
- O servidor de EJBs mais famoso é o JBoss, pois é gratuito, open-source e muito moderno;
- Podemos acessar um EJB a partir de qualquer sistema operacional com suporte a TCP/IP;

Camada EJB

- Diversos tipos de EJB's, um para cada situação:

  - ✓ Session Bean: processamento de dados;
  - ✓ Entity Bean: representação de tabelas do banco de dados e cache de dados;
  - ✓ Message-driven Bean: consome mensagens de filas de MQ-Server;
  - ✓ Session Timed Bean: para processamento agendado;

Agenda – Parte Teórica

1. Introdução
2. Vantagens e casos de sucesso
3. J2EE Vs. Microsoft .NET
4. Composição técnica da arquitetura J2EE
5. Camada Web
6. Camada EJB (componentes de negócio)
7. Clustering
8. Modelos de arquiteturas
9. Demonstrações

Clustering

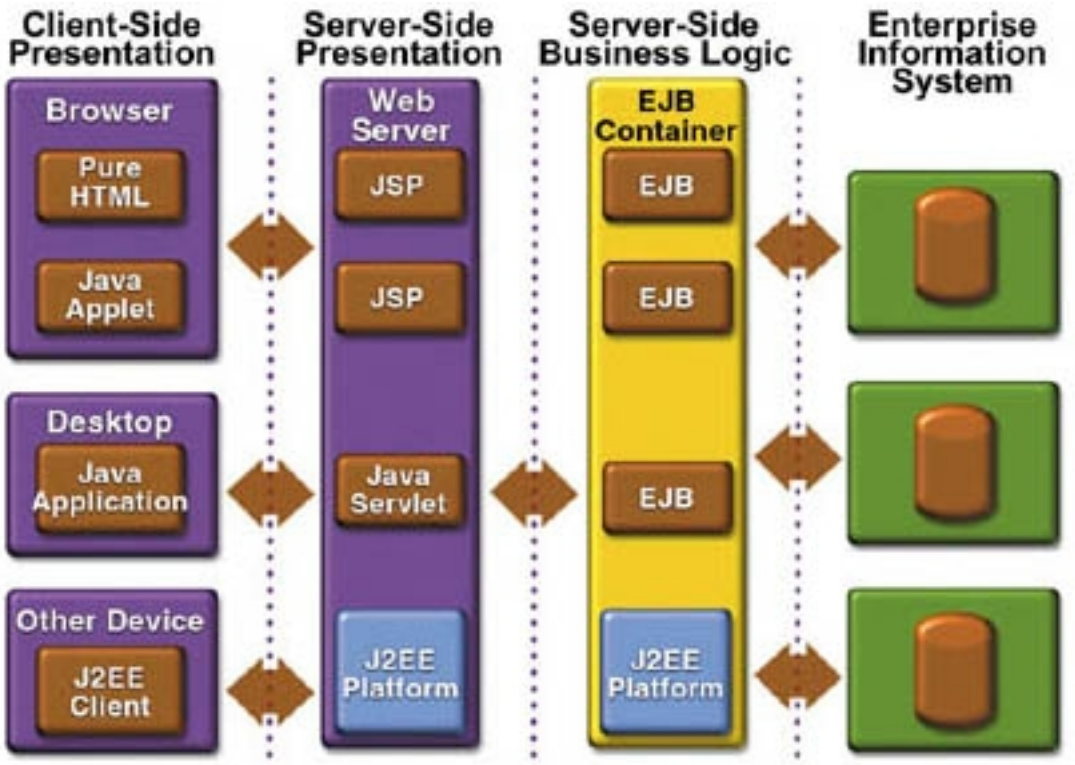
- Cluster = aglomerado, amontoado

- Podemos trabalhar com mais de um servidor por camada:
  - 2 servidores Web
  - 5 servidores EJB
  - 3 servidores RDBMS
  - 8 servidores HTTP

- Quando trabalhamos com cluster, tipicamente os servidores dividem a carga de trabalho através de balanceamento de carga;

Clustering

- Podemos “clusterizar” em qualquer camada para aumentar a capacidade, confiabilidade e disponibilidade



Clustering

- Representa uma forma eficiente e flexível para aumentar a capacidade, disponibilidade e confiabilidade da solução;

- Servidores J2EE open-source disponibilizam sistema de cluster:
  - Apache Tomcat 5.x
  - JBoss 3.x

- Servidores J2EE comerciais:
  - IBM WebSphere
  - BEA WebLogic

Agenda – Parte Teórica

1. Introdução
2. Vantagens e casos de sucesso
3. J2EE Vs. Microsoft .NET
4. Composição técnica da arquitetura J2EE
5. Camada Web
6. Camada EJB (componentes de negócio)
7. Clustering
8. Modelos de arquiteturas
9. Demonstrações

Introdução a Arquiteturas

- J2EE representa uma plataforma flexível para o desenvolvimento de softwares em múltiplas camadas de QUALQUER porte;

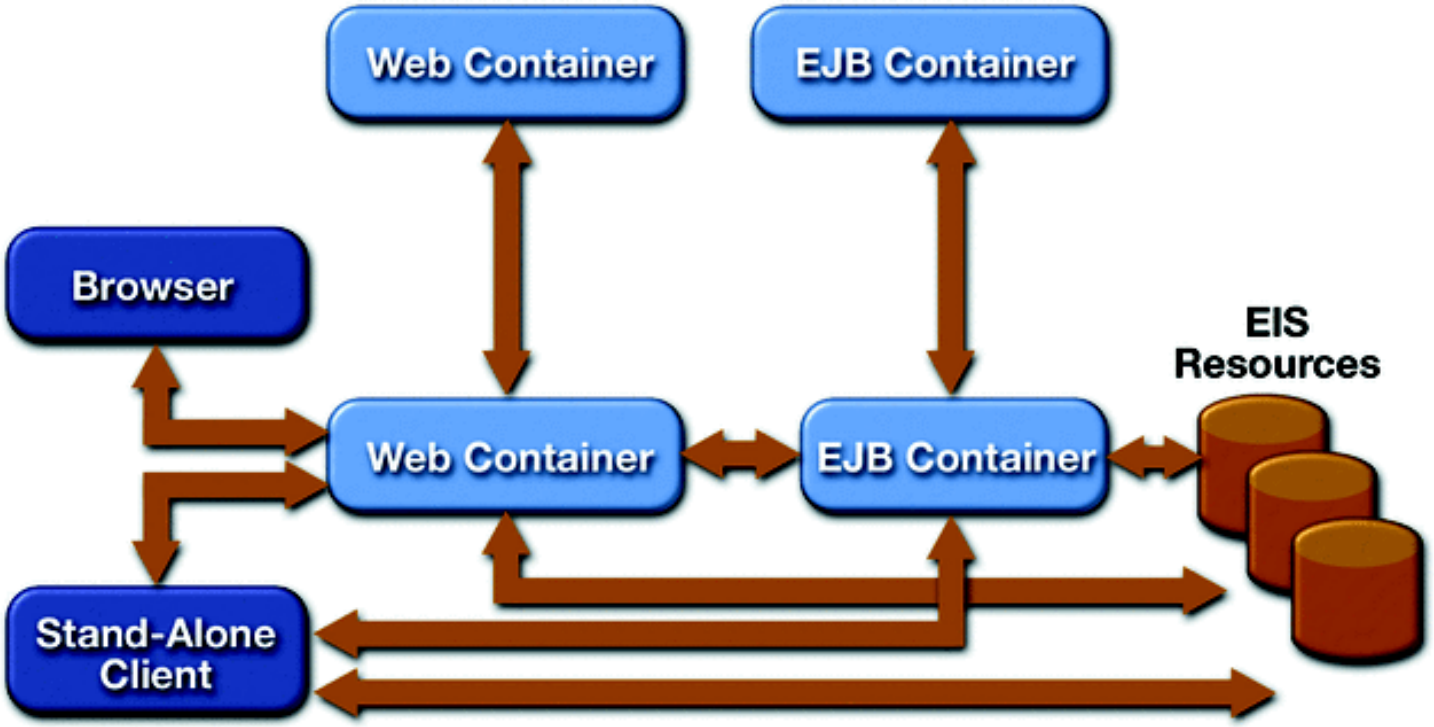
- Oferece diversos recursos: Serlvets, JSP's, Tag Library, Filter, segurança, WebService, transações, persistência, pool de conexões e objetos, entre outros;

- Podemos aplicar design-patterns, melhores práticas e convenções para aumentarmos a qualidade da solução;

Introdução a Arquiteturas

- A definição de uma arquitetura J2EE deve reunir:
  - Tipos de componentes utilizados: Servlet, JSP, Custom Tag, Servlet Filter, EJB, Entity CMP, BMP, assíncrono, transações etc.;
  - Design-patterns e frameworks adotados;
  - Melhores práticas, convenções e métricas de desenvolvimento;
  - Diagramas UML de referência: use-cases, seqüência, classes, componentes e deployment;

Introdução a Arquiteturas



Introdução a Arquiteturas

- Uma **arquitetura** deve **atender** aos requisitos **funcionais** e **não-funcionais** do negócio:
  - Deve atender todas as **funcionalidades esperadas** pelos **usuários**;
  - Deve ser compatível com o **know-how da equipe** de desenvolvimento (interna ou fábrica externa);
  - **Requisitos não-funcionais** devem ser considerados: usuários simultaneos ou requisições por segundo, flexibilidade, segurança, extensibilidade, capacidade de crescimento, disponibilidade desejada, entre outros;

Introdução a Arquiteturas

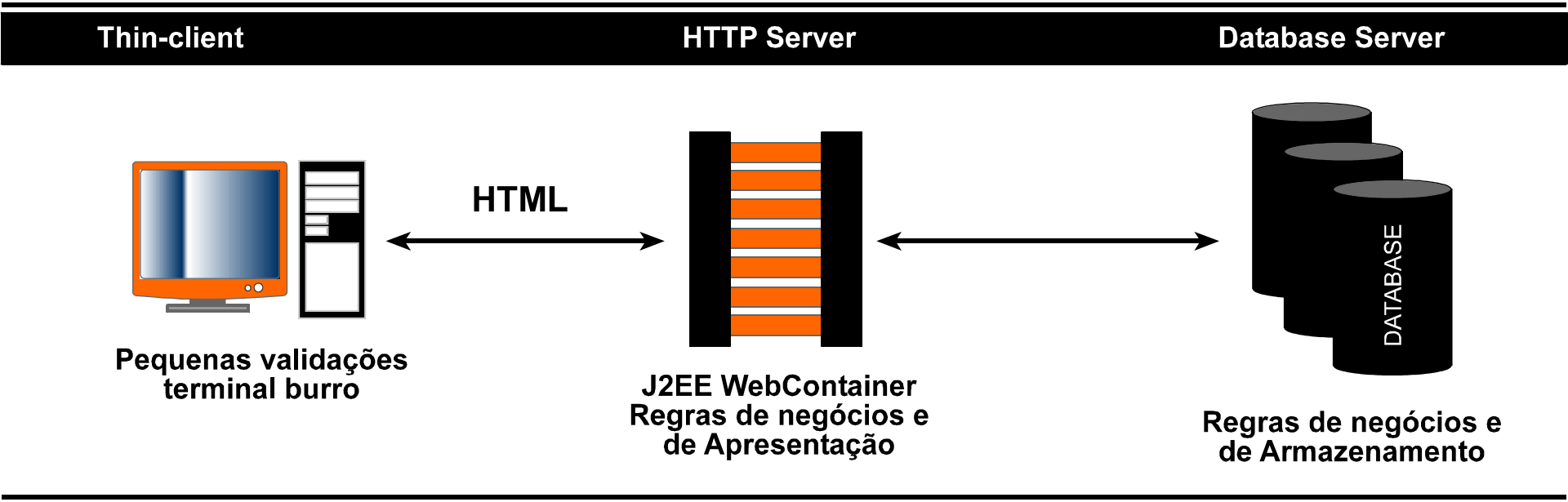
- Considere os seguintes aspectos de ambiente ao decidir elementos de uma arquitetura J2EE:
  - Capacidade técnica da equipe;
  - Necessidade de integração com demais plataformas;
  - Nível de segurança;
  - “Elasticidade” do negócio;
  - Ferramentas disponíveis;
  - Prazo;
  - Budget;
- “More, is not necessarily better...”, Leonardo Galvão.

Introdução a Arquiteturas

- Melhores práticas de arquitetura J2EE:
  - Utilize UML intensamente: cada diagrama tem seu valor, fuja dos diagramas mudos;
  - Não deixe a paixão pela tecnologia da equipe tornar sua empresa um centro de testes de API's e frameworks;
  - Sempre que possível, efetue provas de conceitos: estatísticas não existem!
  - É mais difícil manter um padrão de arquitetura, do que definir um, cuide dos processos e metodologia!

J2EE: Arquitetura I

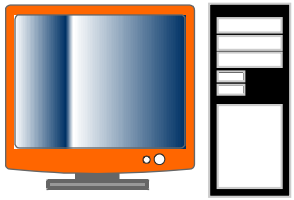
Thin-client, Web-Centric: sem uso de EJB's



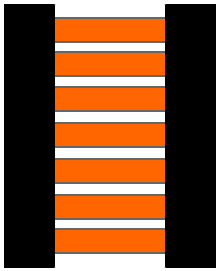
J2EE: Arquitetura II

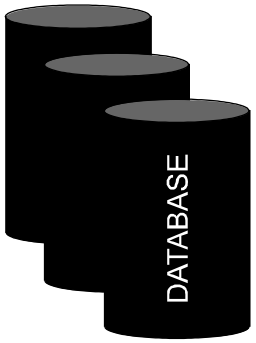
Fat-client, Web-Centric: sem uso de EJB's

Fat-client	HTTP Server	Database Server



XML



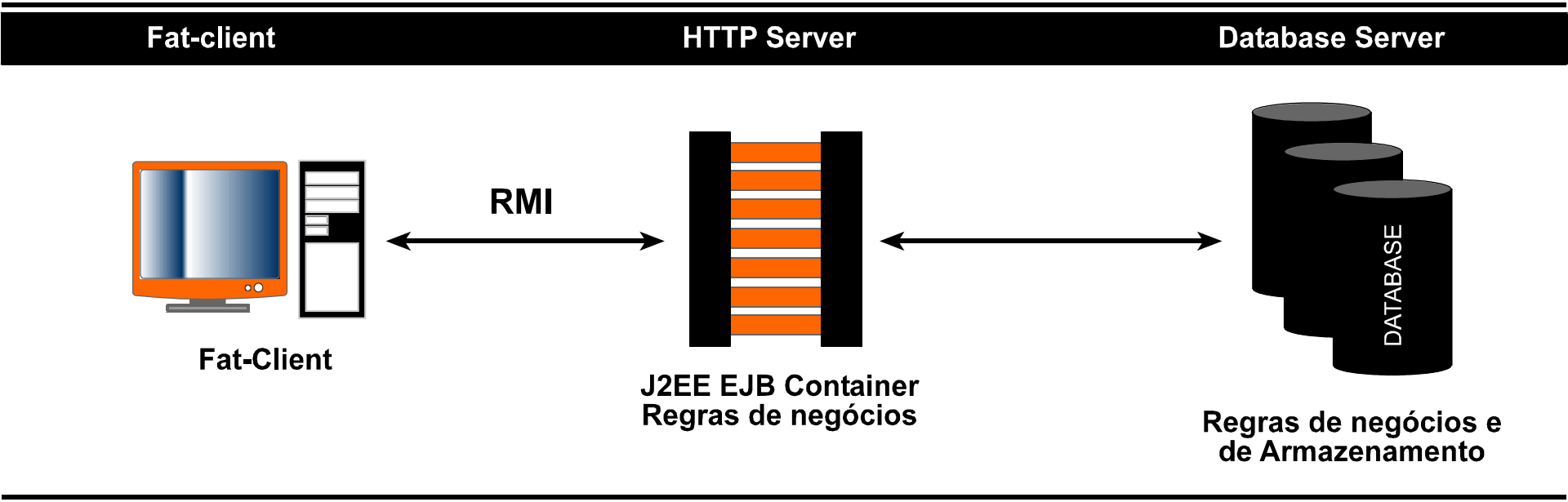


Grande dinâmica na interface
Usuários exigentes:
Swing, Flash, .net entre outros

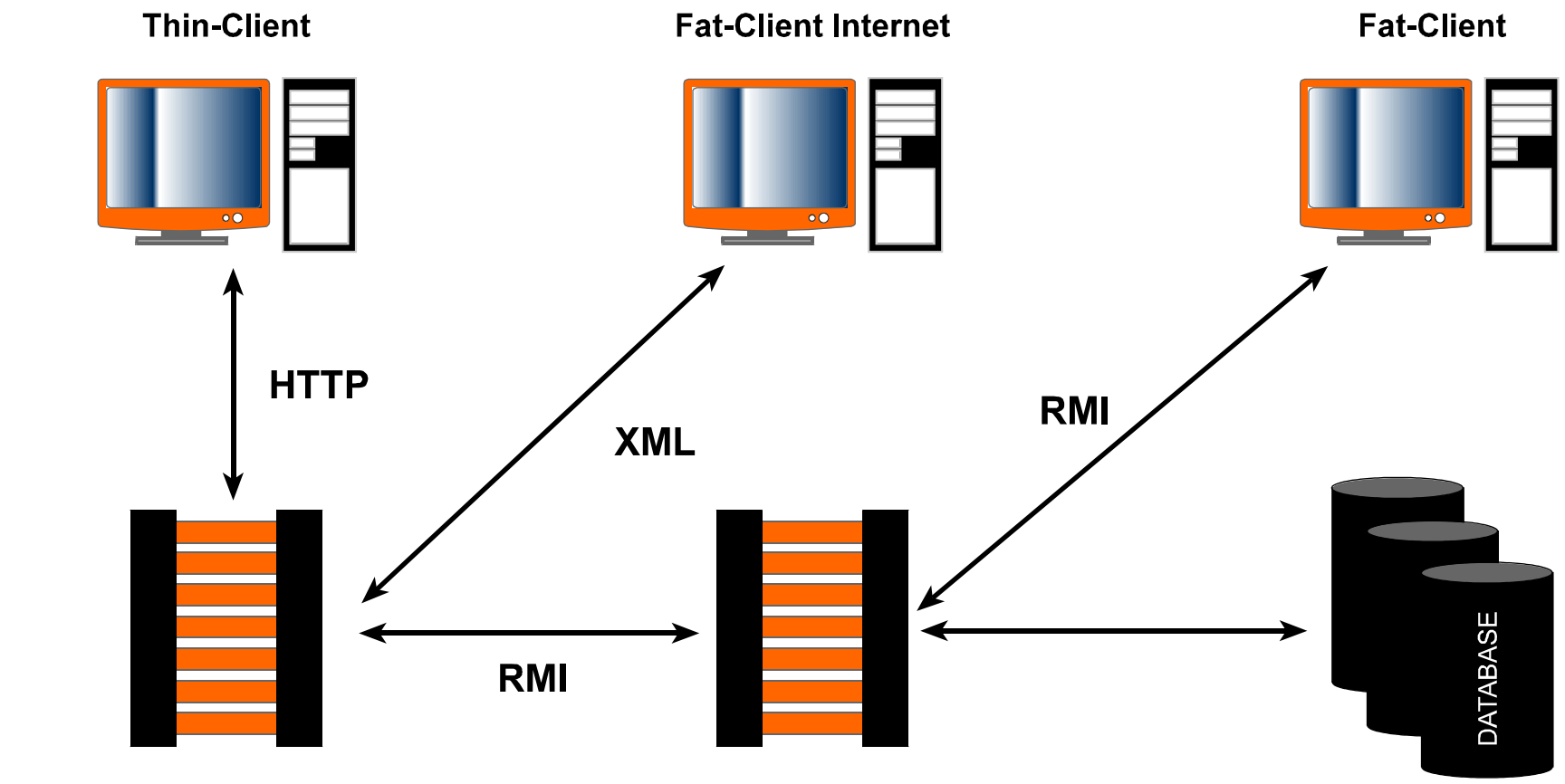
J2EE WebContainer
Regras de negócios e
de Apresentação

Regras de negócios e
de Armazenamento

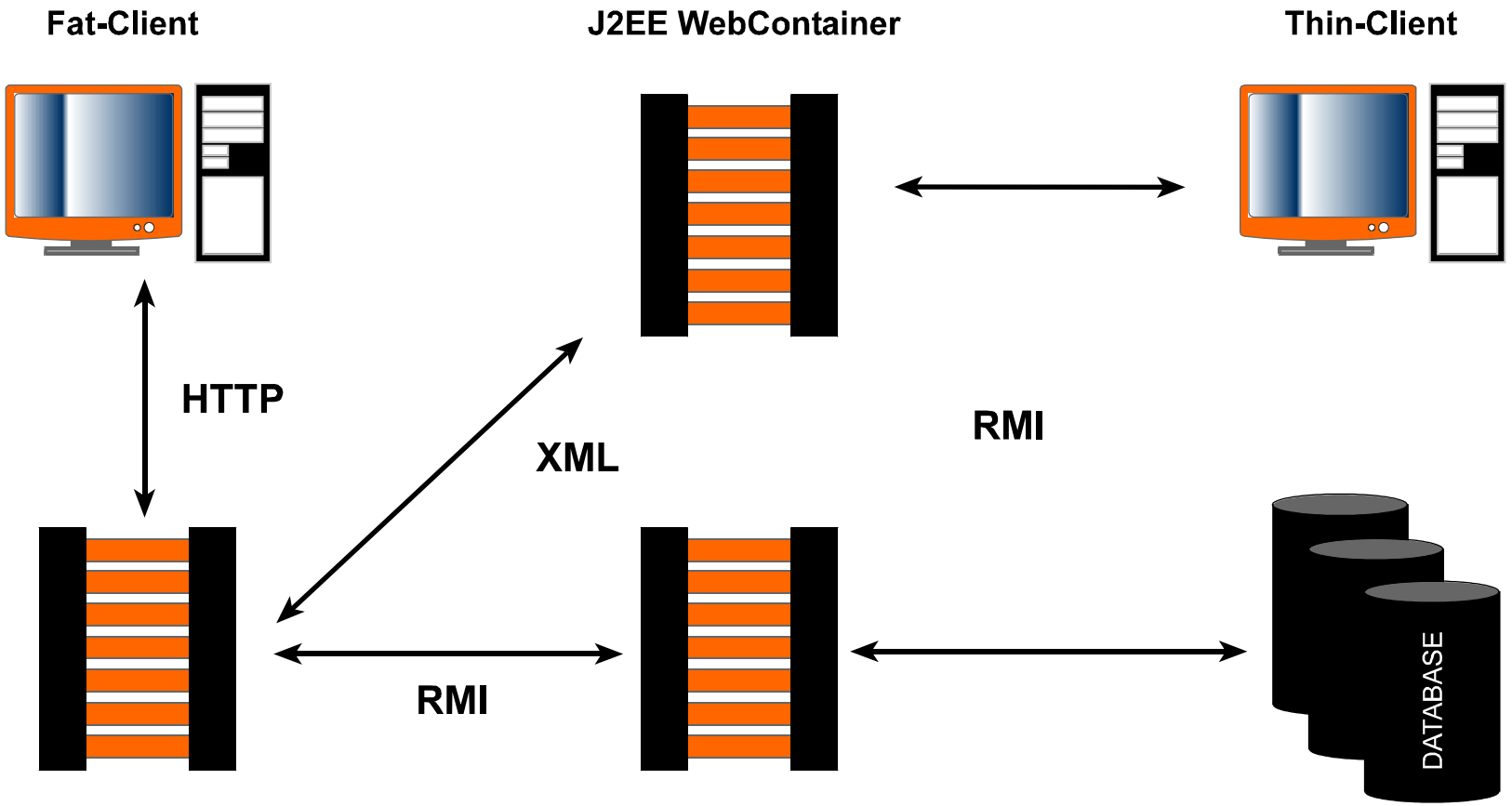
J2EE: Arquitetura III
Fat-client, EJB-Centric



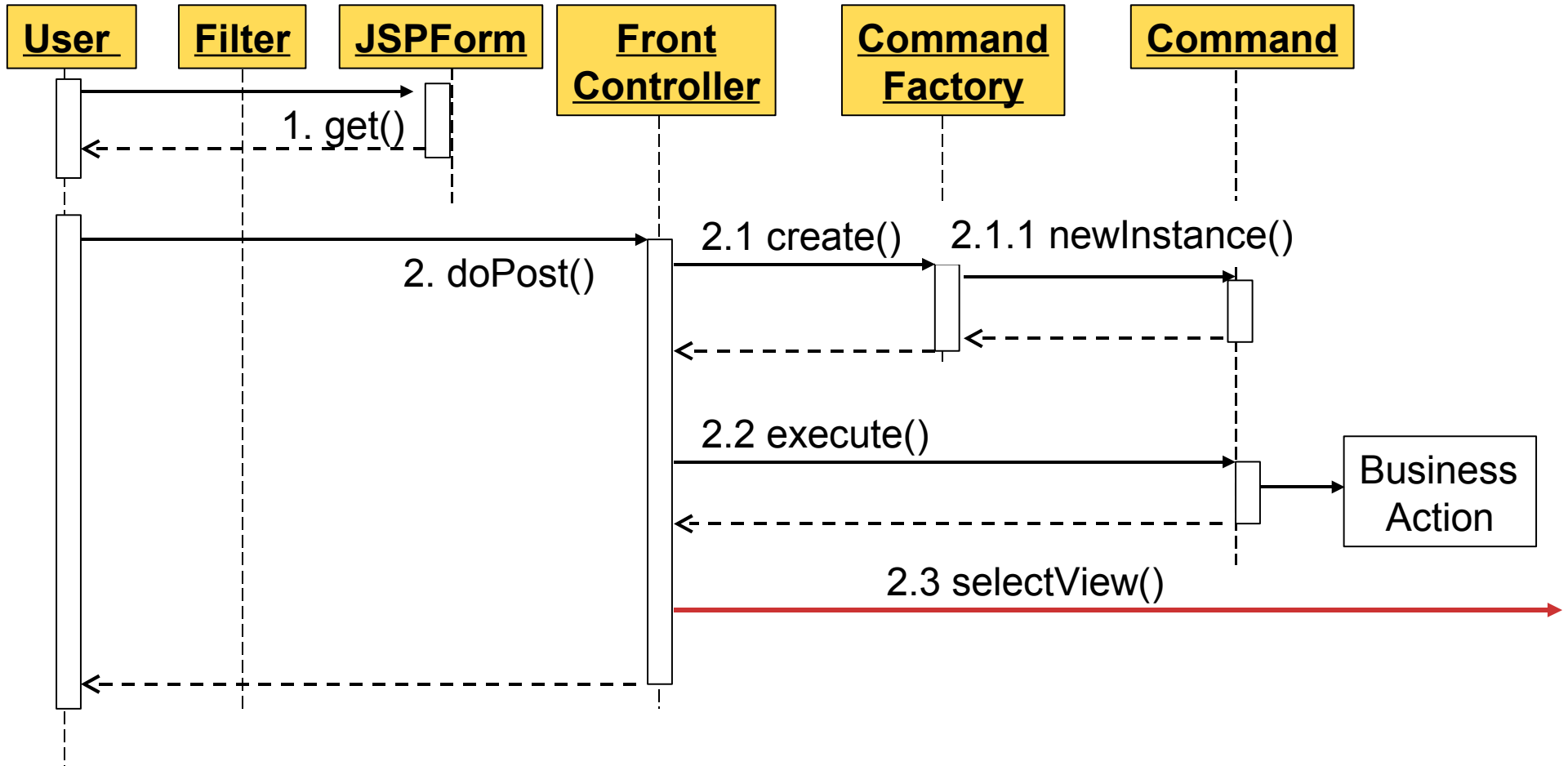
J2EE: Arquitetura IV
Fat-client & Thin-client, EJB-Centric



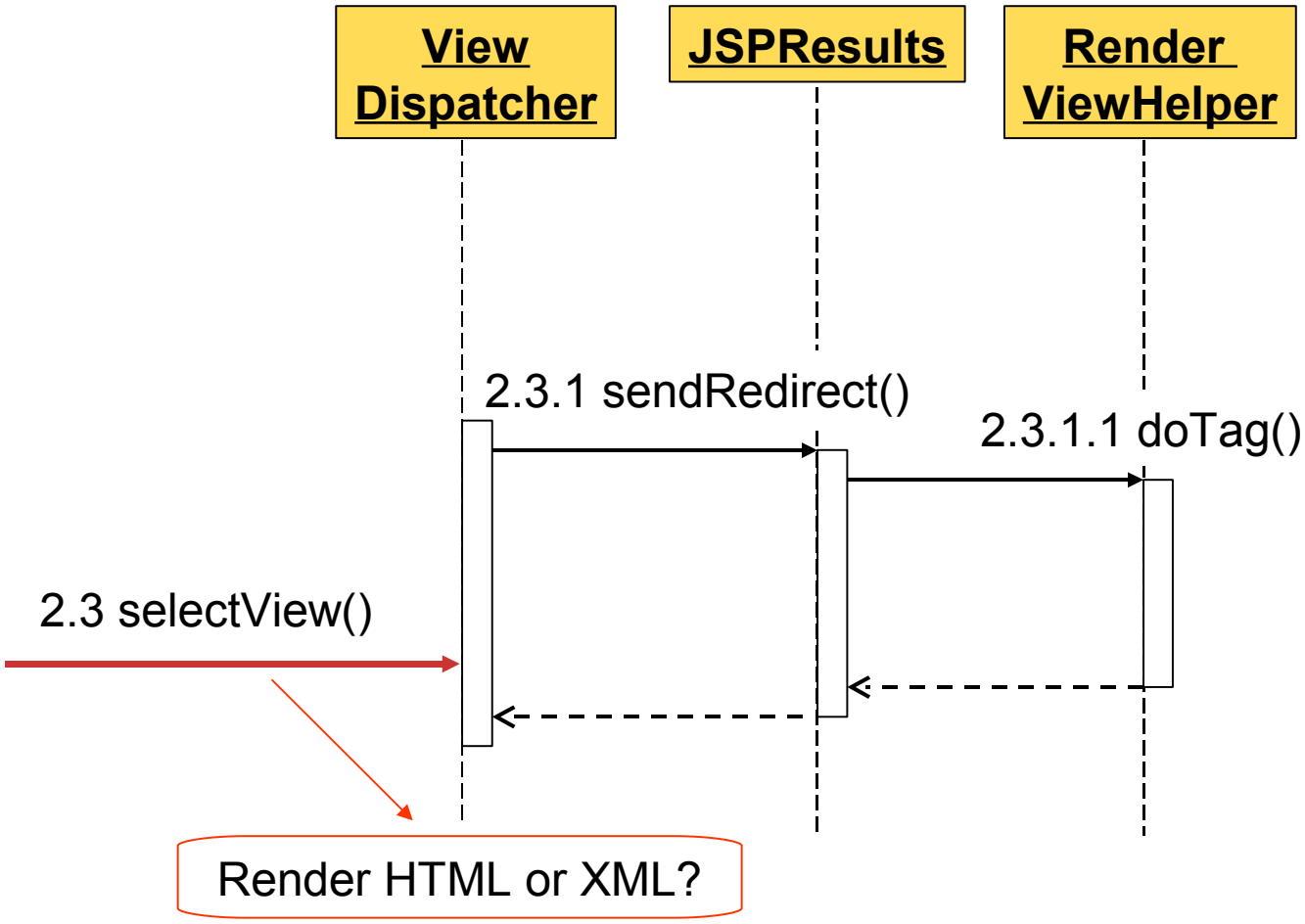
J2EE: Arquitetura V
Baseada em mensagens



Exemplo Arquitetura Web



Exemplo Arquitetura Web



Agenda – Parte Teórica

1. Introdução
2. Vantagens e casos de sucesso
3. J2EE Vs. Microsoft .NET
4. Composição técnica da arquitetura J2EE
5. Camada Web
6. Camada EJB (componentes de negócio)
7. Clustering
8. Modelos de arquiteturas
9. Demonstrações

DEMO

Conclusões

- ✓ J2EE é uma plataforma madura, robusta e flexível para o desenvolvimento de soluções para pequeno, médio e grande porte.
- ✓ J2EE não é proprietário.
- ✓ J2EE oferece opções open-source e pagas.
- ✓ J2EE pode ser utilizado para aplicativos Web, processos de integração e também soluções de alta criticidade.
- ✓ Exemplos, códigos e apostilas: www.globalcode.com.br