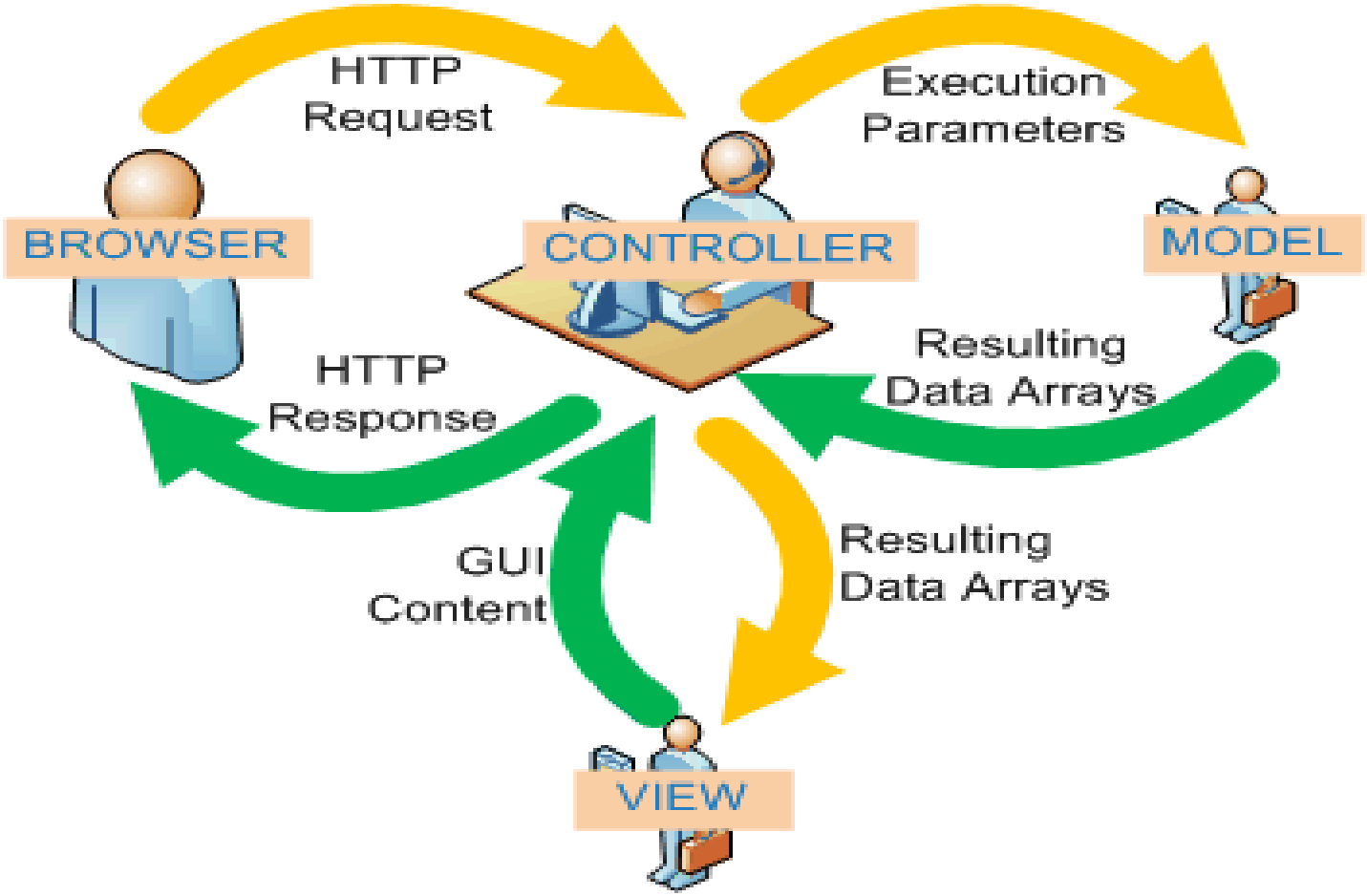
Мрежово програмиране

Архитектура на Web системи

доц. д-р Йордан Денев
denev@fmi.uni-sofia.bg

Model View Controller (MVC)



❑ Contoller

- Declare special environment setting
- Start Session;
- Include configuration settings, database abstraction class, and any other common class.
- Authenticate user, and assign user roles by calling Model object.
- Verify HTTP Request; aggregate and assemble execution parameters from HTTP Request.
- Code a control flow with the help of execution parameters and call Model objects and View objects in the order to facilitate data-flow.

❑ Model

Model classes are written to manage and to regulate the flow of data during the execution of business logic in the application. These classes are instantiated in the Controller file, and are invoked by public methods therein.

There is a connection to the data base engine.

❑ View

View objects (set of files, and not object technically) are basically HTML files containing for example PHP/JSP print statements or simple conditional and loop statements spread within the HTML code.

Ruby on Rails

Ruby on Rails, often shortened to **Rails** or **RoR**, is an open source web application framework for the Ruby programming language.

Rails uses the Model-View-Controller (MVC) architecture pattern to organize application programming.[5]

Ruby on Rails includes tools that make common development tasks easier "out of the box", such as scaffolding that can automatically construct some of the models and views needed for a basic website.[6] Also included are (WEBrick), a simple ruby web server and Rake, a build system. Together with Rails these tools provide a basic development environment.

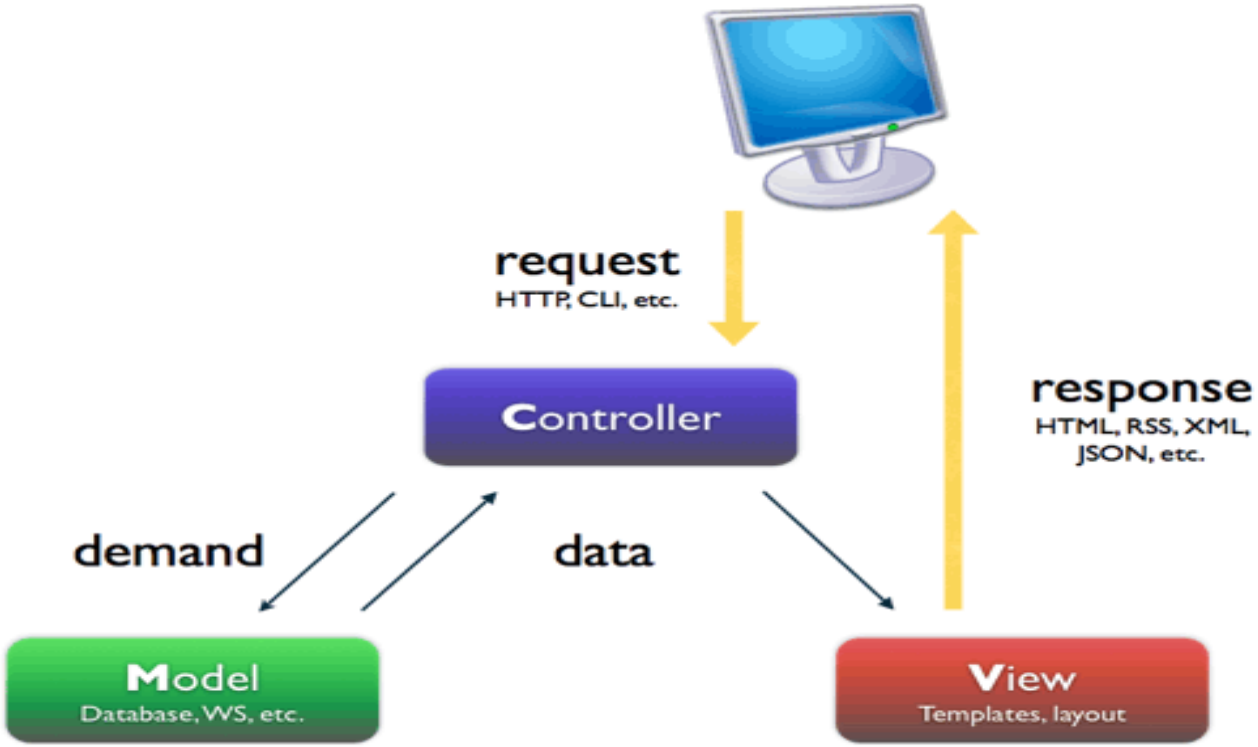
PHP MVC frameworks

- Akelos PHP Framework a Ruby on Rails port to PHP4/5.
- Barebonesmvc A one-file, no-configuration, PHP 5 MVC framework.
- CakePHP webapplication framework modeled after the concepts of Ruby on Rails.
- CodeIgniter A PHP MVC framework.
- ash.MVC A Simple MVC Framework with PHP.
- FUSE A powerful but easy-to-use PHP 5 Framework for MVC development modeled after the concepts of Ruby on Rails.
- Zend Framework A PHP 5-based MVC framework modeled after the concepts of Ruby on Rails.
- Symfony Framework PHP 5 MVC Framework modeled after the concepts of Ruby on Rails.

❑The Akelos PHP Workflow

1. Akelos will break up your request into three parameters according to your /config/routes.php file (more on this later)
   - controller: book
   - action: show
   - id: 2
2. Once Akelos knows about this request it will look for the file /app/controllers/book_controller.php and if found it will instantiate the class BookController
3. The controller will look for a model that matches the parameter controller from the request. In this case it will look for /app/models/book.php. If found, it will create an instance of the model on the controller $this->Book attribute. If an id is on the request, it will search into the database for the Book with the id 2 and that will remain on $this->Book
4. Now it will call the action show from the BookController class if it's available.
5. Once the show action has been executed, the controller will look for the view file at /app/views/book/show.tpl and will render the results into the $content_for_layout variable.
6. Now Akelos will look for a layout named like the controller at /app/views/layouts/book.tpl. If found it will render the layout inserting $content_for_layout content and sending the output to the browser.

Simfony framework

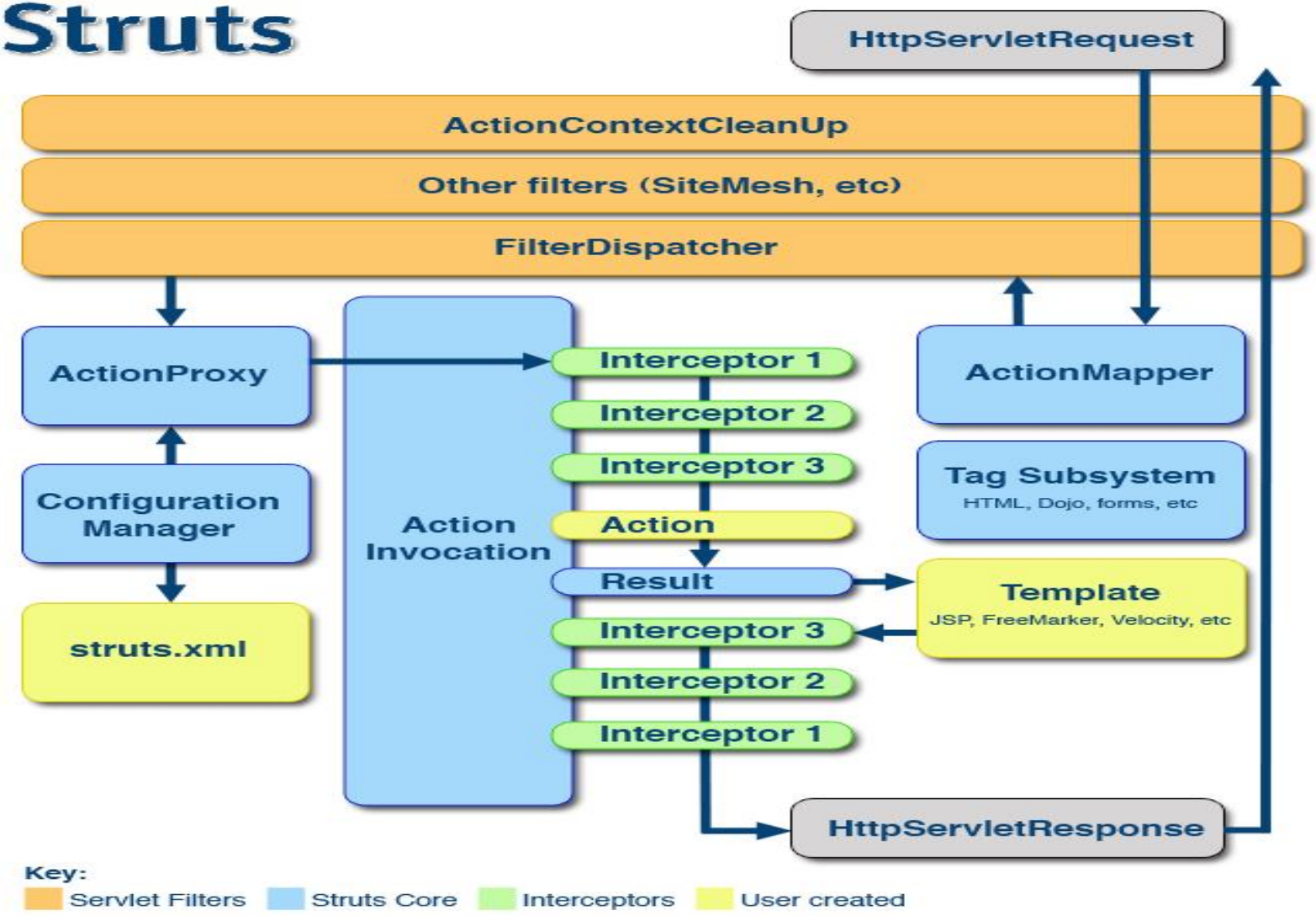


- The **Model** layer defines the business logic (the database belongs to this layer). Symfony stores all the classes and files related to the Model in the lib/model/ directory.
- The **View** is what the user interacts with (a template engine is part of this layer). In symfony, the View layer is mainly made of PHP templates. They are stored in various templates/ directories .
- The **Controller** is a piece of code that calls the Model to get some data that it passes to the View for rendering to the client. So, all requests are managed by front controllers (index.php and frontend_dev.php). These front controllers delegate the real work to **actions**. These actions are logically grouped into **modules**.

MVC Java frameworks

- Induction
- JSF
- LongJump
- Oracle Application Framework
- PureMVC, a framework for Java
- Sofia
- Spring MVC Framework
- Struts
- Stripes
- Tapestry[6]
- WebObjects
- WebWork

Struts


HttpServletRequest
ActionContextCleanUp
Other filters (SiteMesh, etc)
FilterDispatcher
ActionProxy
Configuration Manager
struts.xml
Action Invocation
Interceptor 1
Interceptor 2
Interceptor 3
Action
Result
Interceptor 3
Interceptor 2
Interceptor 1
ActionMapper
Tag Subsystem
HTML, Dojo, forms, etc
Template
JSP, FreeMarker, Velocity, etc
HttpServletResponse
Key:
Servlet Filters
Struts Core
Interceptors
User created

- **The Struts Controller Components:**

  Whenever a user request for something, then the request is handled by the Struts Action Servlet. When the ActionServlet receives the request, it intercepts the URL and based on the Struts Configuration files, it gives the handling of the request to the Action class. Action class is a part of the controller and is responsible for communicating with the model layer.

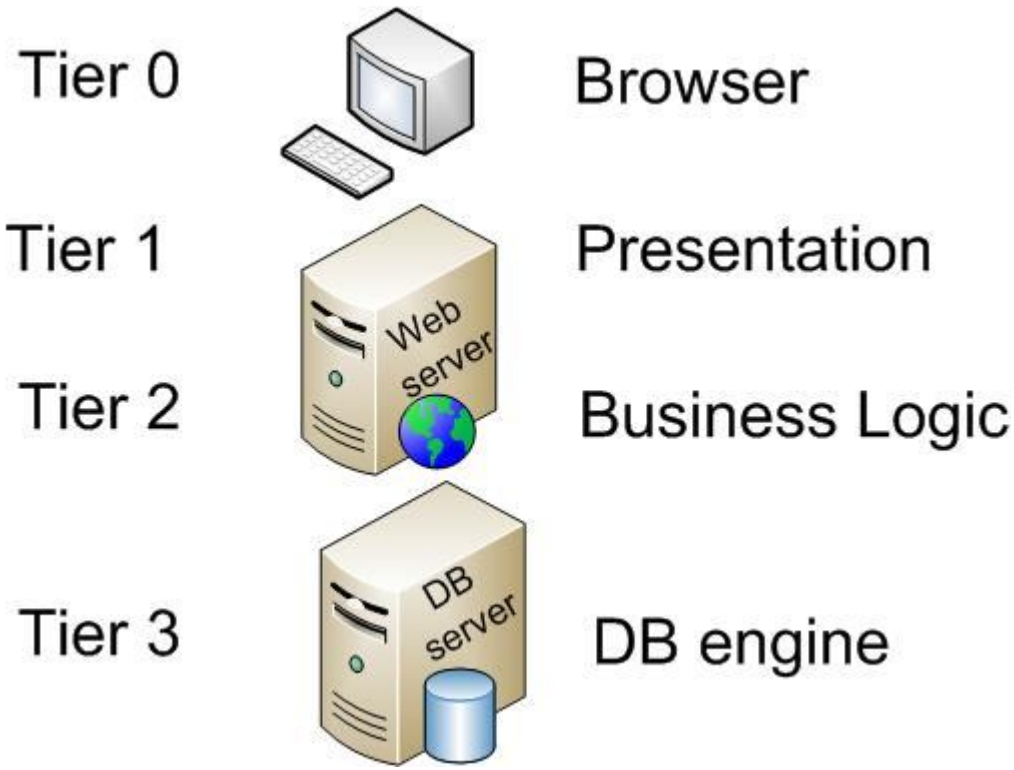
- **The Struts View Components:**

  They are responsible for displaying the information provided by the model components. Mostly we use the Java Server Pages (JSP) for the view presentation. To extend the capability of the view we can use the Custom tags, java script etc.

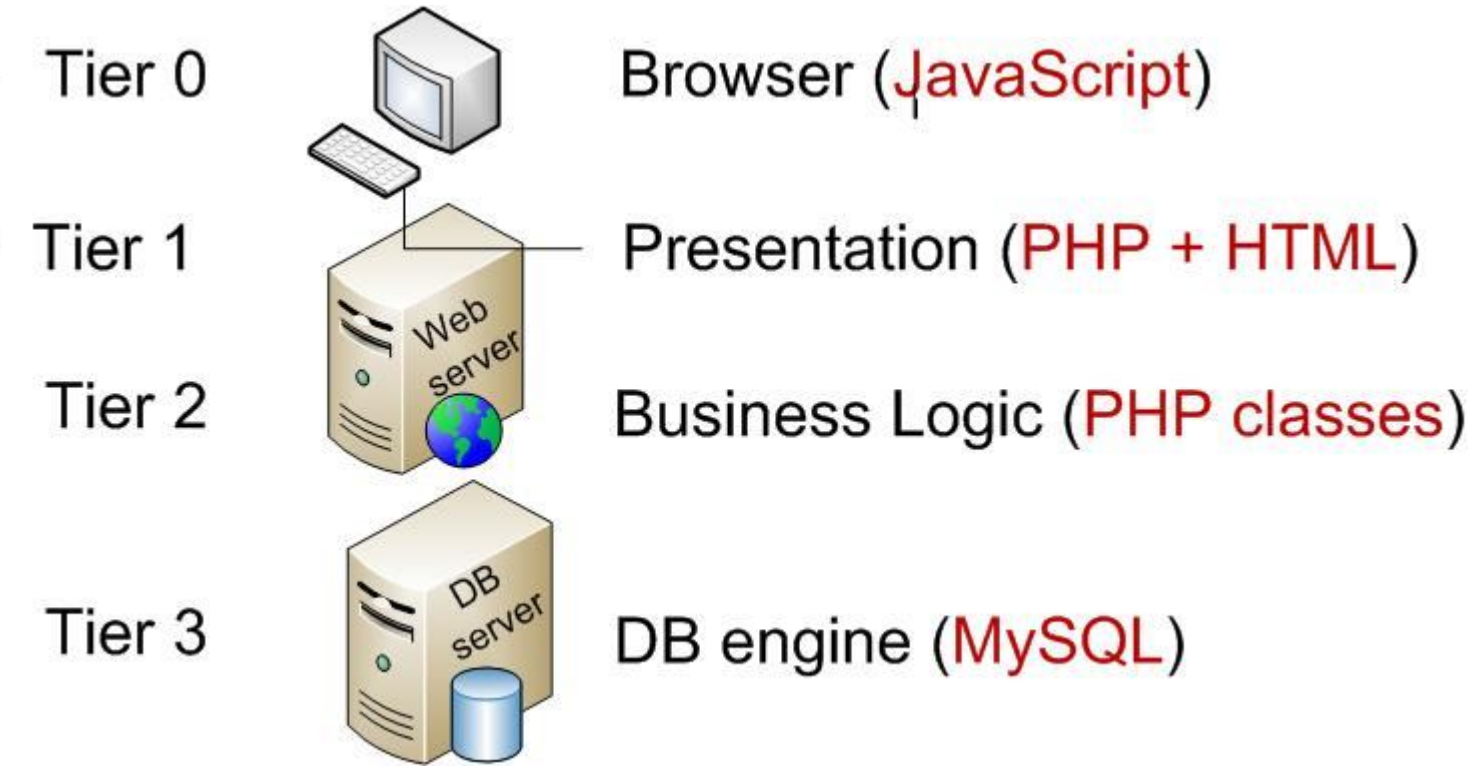
- **The Struts model component:**

  The model components provides a model of the business logic behind a Struts program. It provides interfaces to databases or back- ends systems. Model components are generally a java class. There is not any such defined format for a Model component.

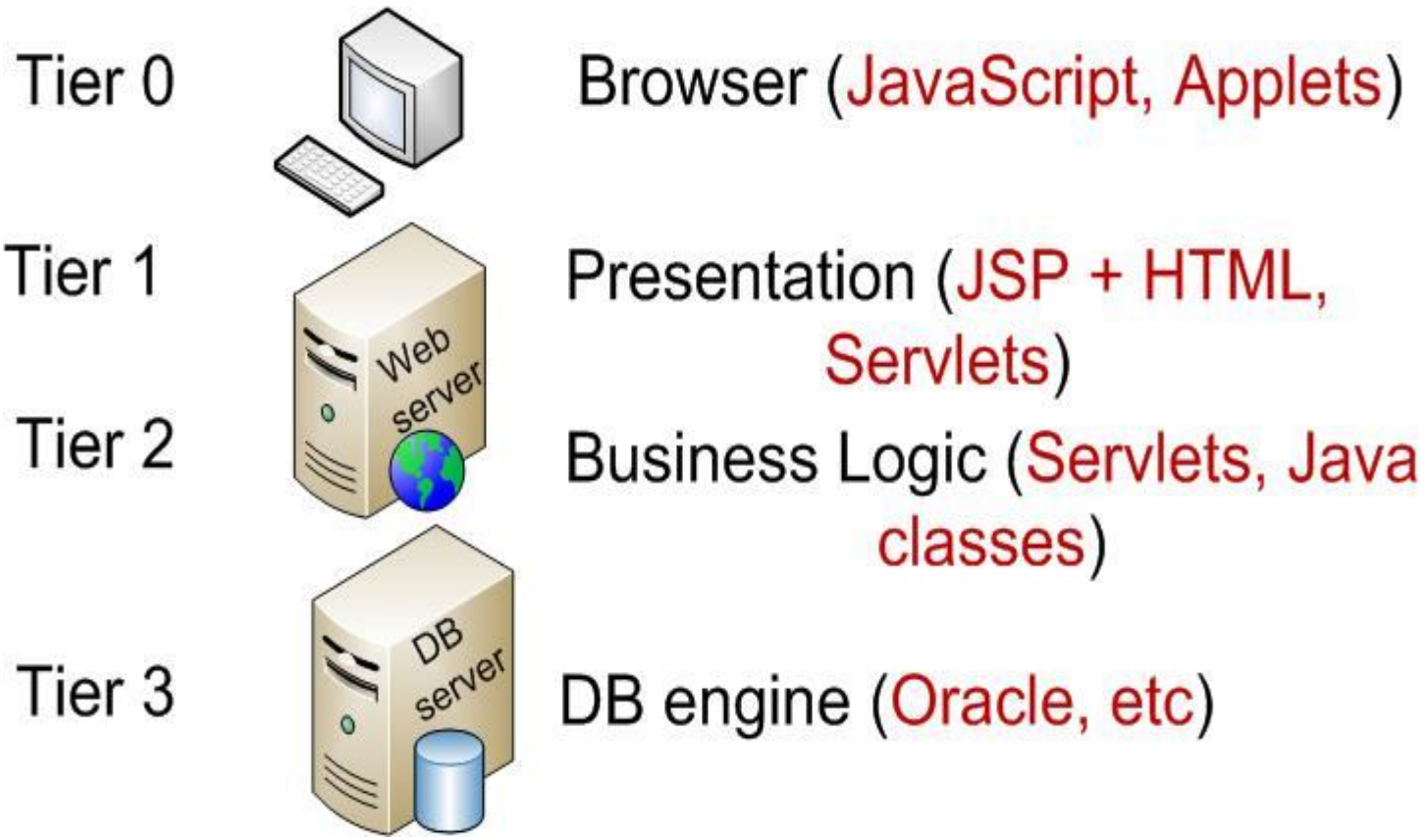
Multitier Architecture



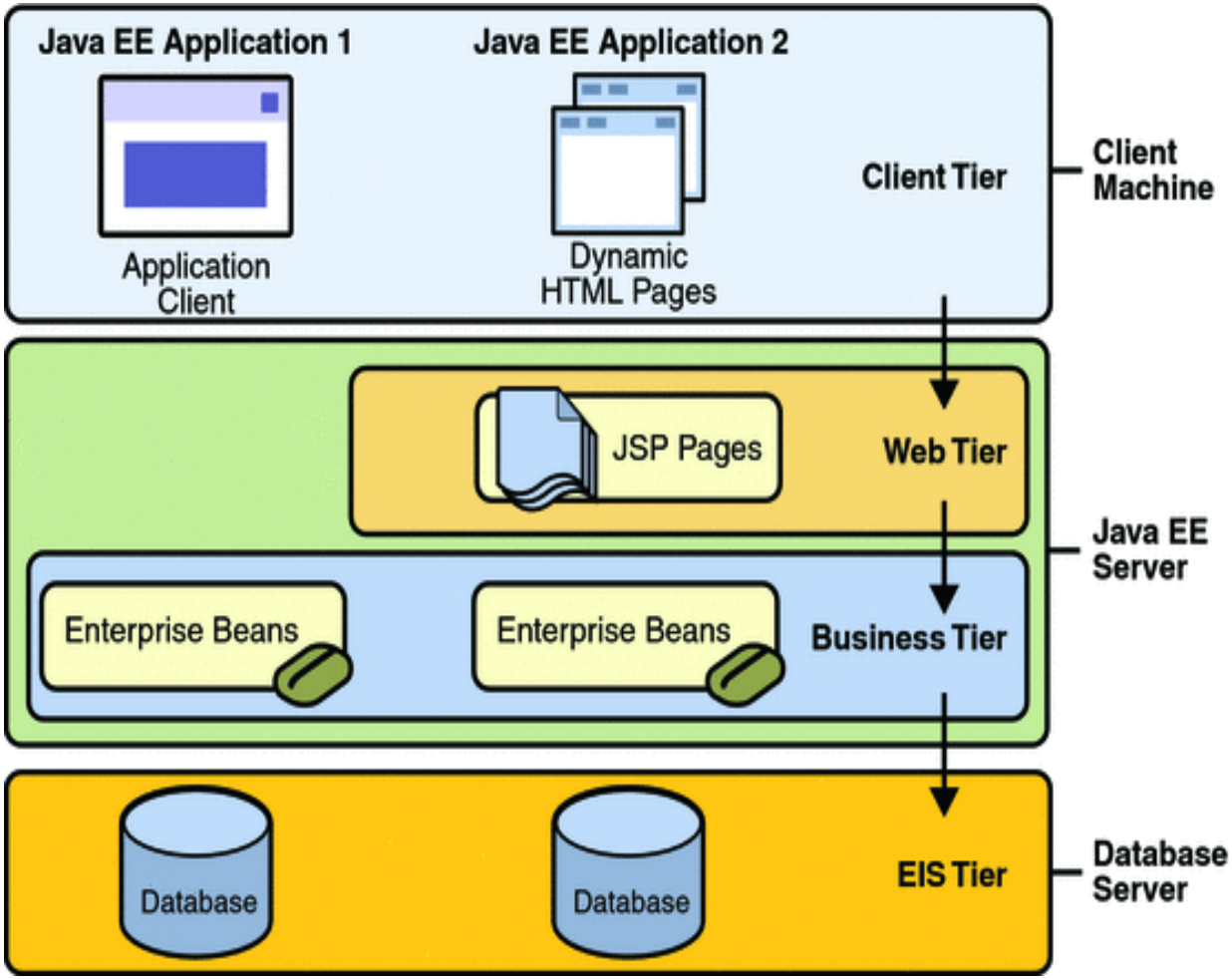
PHP approach



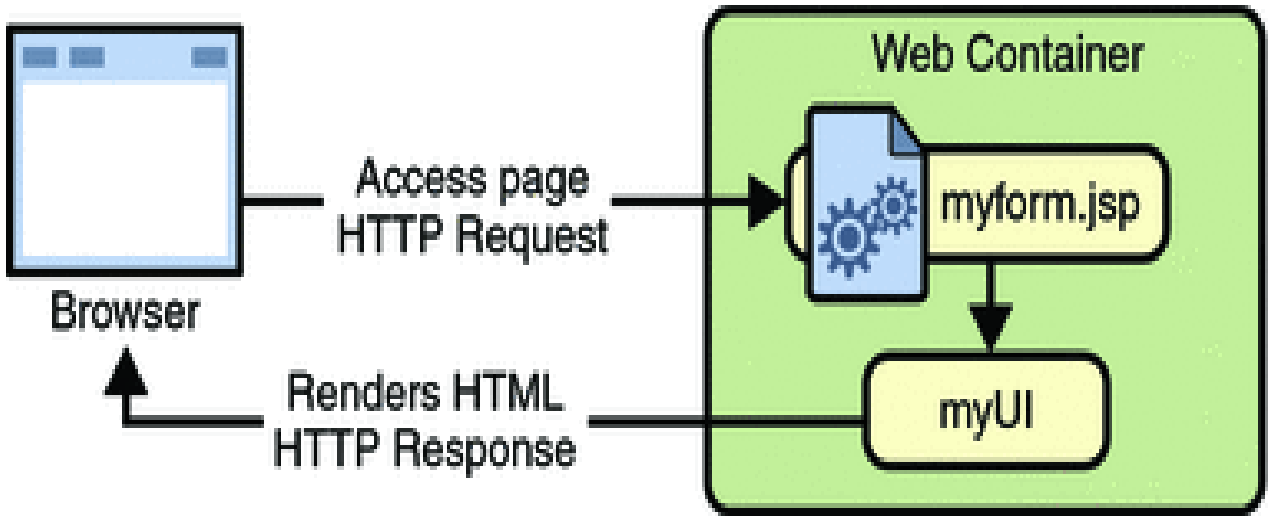
Simple Java Approach



Java EE5 architecture



❑ JavaServer Faces Technology User Interface



The JSP page, myform.jsp, is a **JavaServer Faces page**, which is a JSP page that includes JavaServer Faces tags. It expresses the user interface components by using custom tags defined by JavaServer Faces technology.

Why Enterprise Java Beans (EJB)

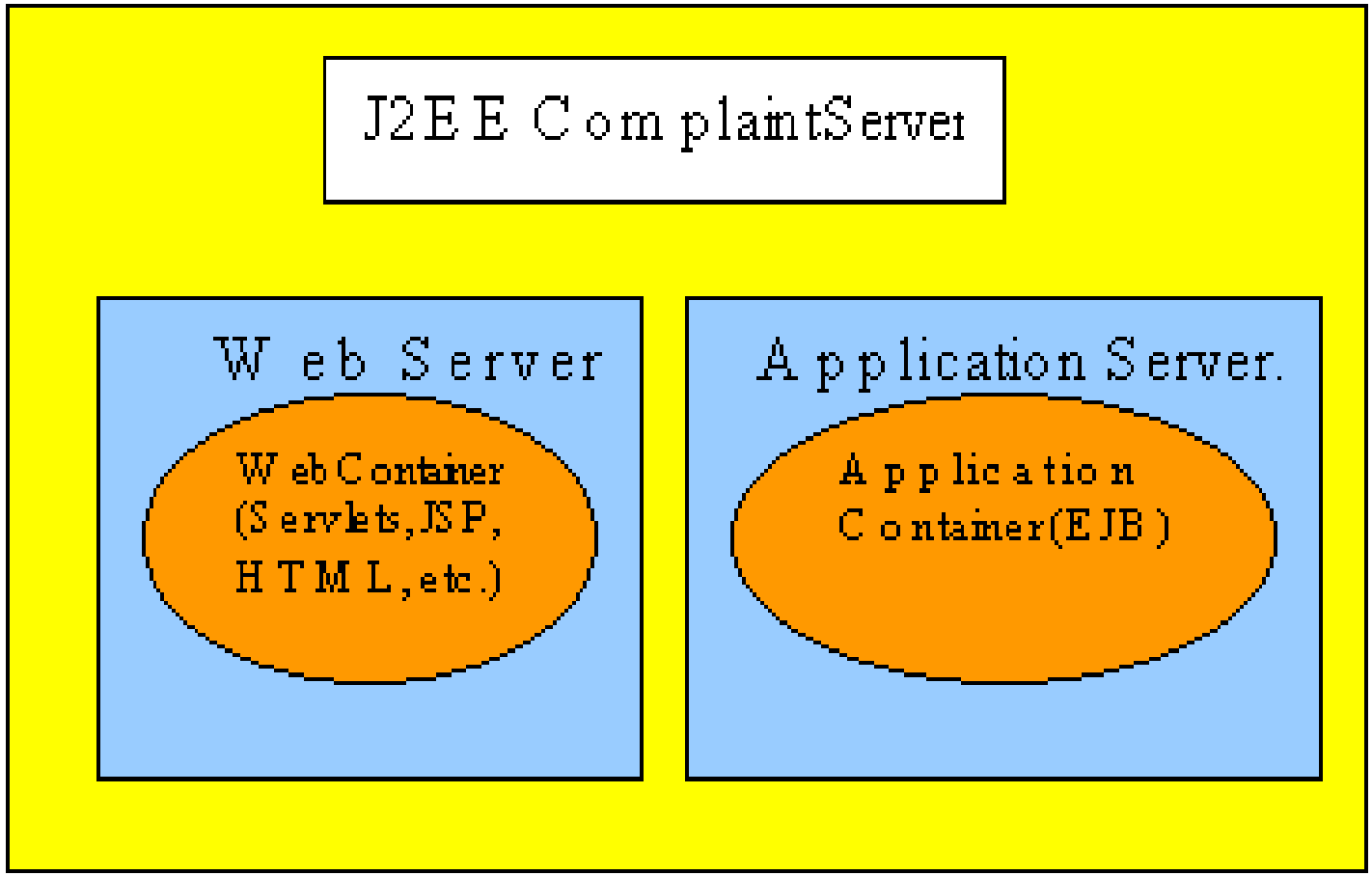
It is a standard way of writing distributed components. Here comes the features that are not available with RMI.

1. Security.
2. Searching .
3. Transactions.
4. Persistence .

J2EE server and EJB container

The J2EE server provides an environment that supports the execution of applications developed using Enterprise JavaBeans™ (EJB) components. It manages and coordinates the allocation of resources to the applications.

The J2EE server must provide one or more EJB containers. An EJB container manages the enterprise beans contained within it. For each enterprise bean, the container is responsible for registering the object, providing a remote interface for the object, creating and destroying object instances, checking security for the object, managing the active state for the object, and coordinating distributed transactions. Optionally, the container can also manage all persistent data within the object.


J2EE ComplaintServer
Web Server
Web Container
(Servlets, JSP,
HTML, etc.)
Application Server.
Application
Container(EJB)

Commercial J2EE servers

- BEA WebLogic Server (BEA Systems)

  It is the industry's most comprehensive Java platform for developing, deploying, and integrating enterprise applications. It provides the foundation for application grid, which is an architecture that enables enterprises to outperform their competitors while minimizing operational costs

- WebSphere Application Server (IBM)

  Java EE 5 certification, EJB 3.0 support and Java Persistence API (JPA) and Java Development Kit (JDK) 6.0, deliver simplified programming models for building reusable persistent object

- Netscape Application Server (Netscape)
- Oracle Application Server

  Oracle has acquired BEA Systems, Inc., a leading provider of enterprise application infrastructure solutions.

Open source J2EE servers

- **Apache Geronimo** is a new effort coordinated by the Apache Software Foundation to make a J2EE compatible container.
- **JBoss** is advanced middleware with a full J2EE based personality that IT departments look for. But that is not all, the OEM and ISV community embraced JBoss as a highly flexible service oriented architecture on which to build their own products.
- **JFox** is an Open Source, standards-compliant, J2EE based application server implemented in 100% Pure Java.
- J**OnAS** is the Open Source implementation by ObjectWeb of the J2EETM specification. JOnAS is a pure JavaTM implementation of this specification that relies on the JDK. JOnAS is part of the ObjectWeb Open Source initiative, which was launched in collaboration with several partners including Bull, the France Telecom R&D division and INRIA.
- **OpenEJB** is an open source, modular, configurable, and extendable EJB Container System and EJB Server. It comes with fast, lightweight EJB Servers for both Local and Remote access. That's right, deploy your EJBs into the container system, then just start the Remote EJB Server from the command line! Or, put OpenEJB in your class path and use it as an embedded library through the Local EJB Server.

❑ Java Enterprise Beans

(according http://download.oracle.com/javaee/5/tutorial/doc)

Types

- Session Beans
  - Stateful beans - they keep track of which calling program they are dealing with throughout a session
  - Stateless Beans - are distributed objects that do not have state associated with them thus allowing concurrent access to the bean
- Entity Beans that managed persistence.
- Message Driven Beans - entirely new type of bean designed to handle asynchronous JMS messages

❑ A remote access in EE5

A remote client of an enterprise bean has the following traits:

- It can run on a different machine and a different Java virtual machine (JVM) than the enterprise bean it accesses. (It is not required to run on a different JVM.)
- It can be a web component, an application client, or another enterprise bean.
- To a remote client, the location of the enterprise bean is transparent.

To create an enterprise bean that allows remote access, you must do one of the following:

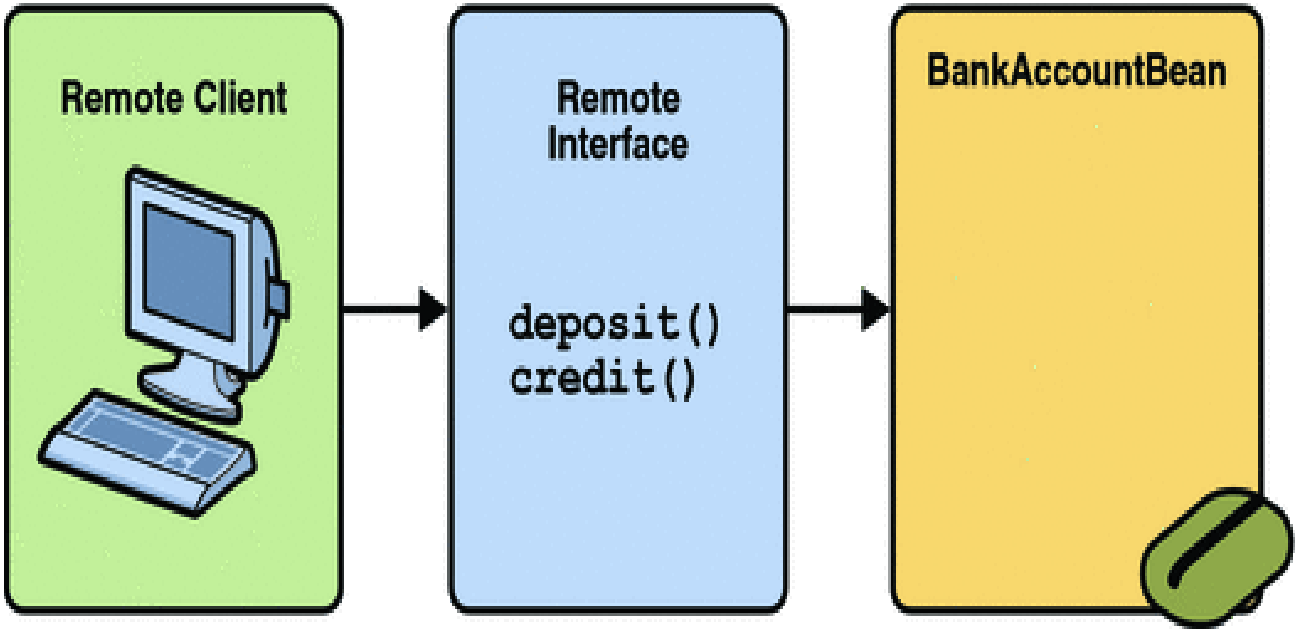
- Decorate the business interface of the enterprise bean with the @Remote annotation:

```
@Remote
public interface InterfaceName { ... }
```

- Decorate the bean class with @Remote, specifying the business interface or interfaces:

```
@Remote(InterfaceName.class)
public class BeanName implements InterfaceName {
  ... }
```

The **remote interface** defines the business and life cycle methods that are specific to the bean.



❑ Local Clients in EE5

A local client has these characteristics:

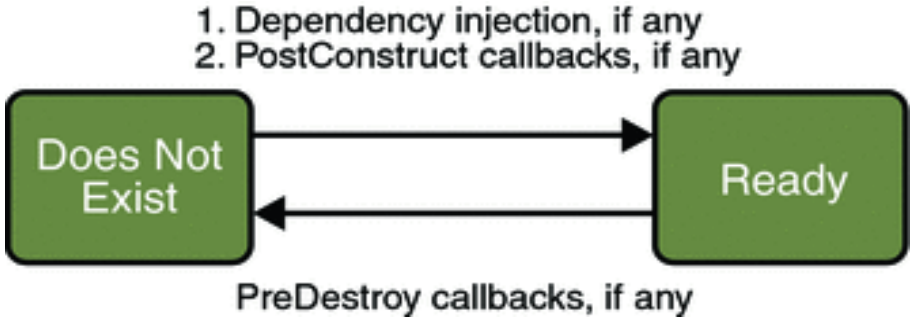
- It must run in the same JVM as the enterprise bean it accesses.
- It can be a web component or another enterprise bean.
- To the local client, the location of the enterprise bean it accesses is not transparent.

- The **local business interface** defines the bean's business and life cycle methods. If the bean's business interface is not decorated with @Local or @Remote, and the bean class does not specify the interface using @Local or @Remote, the business interface is by default a local interface. To build an enterprise bean that allows only local access, you may, but are not required to do one of the following:
  - Annotate the business interface of the enterprise bean as a @Local interface.
  - Specify the interface by decorating the bean class with @Local and specify the interface name.

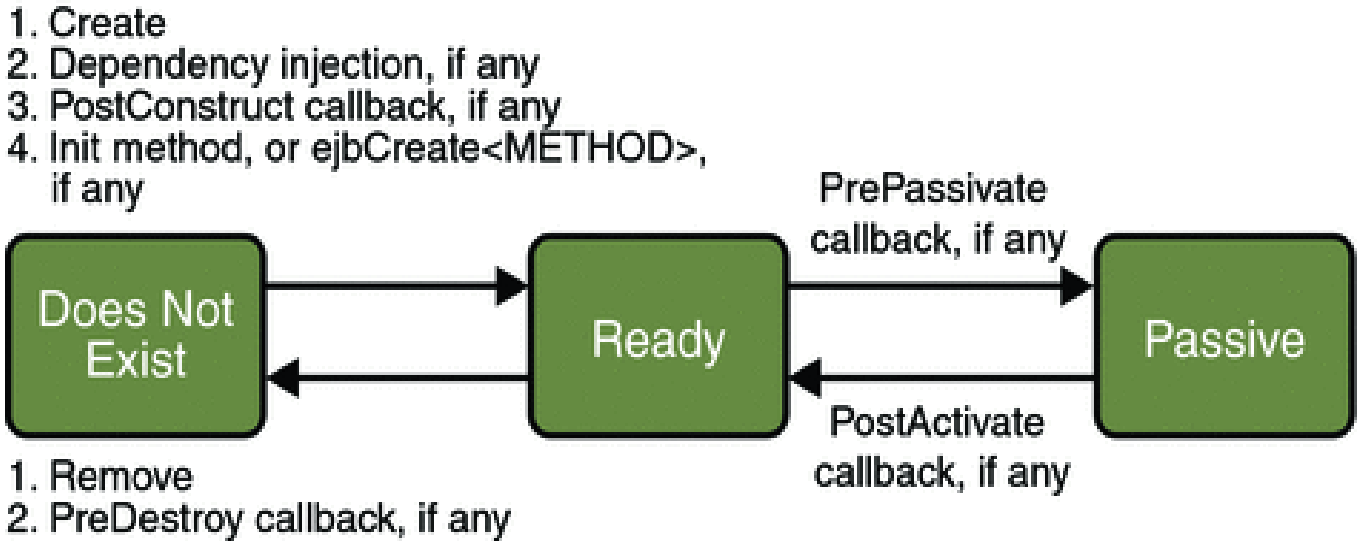
❑Deciding on Remote or Local Access

- **Tight or loose coupling of related beans**:. Tightly coupled beans are good candidates for local access.
- **Type of client**: If an enterprise bean is accessed by application clients, then it should allow remote access.
- **Component distribution**: In a distributed scenario, the enterprise beans should allow remote access.
- **Performance.**

- The life cycle of stateless bean



- The life cycle of stateful bean



❑ An example of a stateless bean

The purpose of *converter i*s to calculate currency conversions between Japanese yen and Eurodollars.

```
package com.sun.tutorial.javaee.ejb;
import java.math.BigDecimal;
import javax.ejb.Remote;
@Remote
public interface Converter {
  public BigDecimal dollarToYen(BigDecimal dollars);
  public BigDecimal yenToEuro(BigDecimal yen);
}
```

```
package com.sun.tutorial.javaee.ejb;
import java.math.BigDecimal;
import javax.ejb.*;
@Stateless
public class ConverterBean implements Converter {
   private BigDecimal yenRate = new BigDecimal("115.3100");
   private BigDecimal euroRate = new BigDecimal("0.0071");

   public BigDecimal dollarToYen(BigDecimal dollars) {
    BigDecimal result = dollars.multiply(yenRate);
    return result.setScale(2, BigDecimal.ROUND_UP);
   }

   public BigDecimal yenToEuro(BigDecimal yen) {
   BigDecimal result = yen.multiply(euroRate);
   return result.setScale(2, BigDecimal.ROUND_UP);
   }
}
```

- For an application client look in

http://download.oracle.com/javaee/5/tutorial/doc/bnbnj.html

- For a Web client look in

http://download.oracle.com/javaee/5/tutorial/doc/bnbnp.html

- How to compile look in

http://download.oracle.com/javaee/5/tutorial/doc/bnbnc.html