Java :: Real Life

Eugene Dzhurinsky

March 9, 2012

Thinking in Object-Oriented way

Benefits of encapsulation

- Keep code and data together.
- Single point of modification.
- One class - one responsibility.
- Easy maintenance.
- Unit tests.

Thinking in Object-Oriented way

Benefits of encapsulation

- Keep code and data together.
- Single point of modification.
- One class - one responsibility.
- Easy maintenance.
- Unit tests.

Thinking in Object-Oriented way

Benefits of encapsulation

- Keep code and data together.
- Single point of modification.
- One class - one responsibility.
- Easy maintenance.
- Unit tests.

Thinking in Object-Oriented way

Benefits of encapsulation

- Keep code and data together.
- Single point of modification.
- One class - one responsibility.
- Easy maintenance.
- Unit tests.

Thinking in Object-Oriented way

Benefits of encapsulation

- Keep code and data together.
- Single point of modification.
- One class - one responsibility.
- Easy maintenance.
- Unit tests.

Thinking in Object-Oriented way

Prefer composition over inheritance. Is-a versus has-a principle.

- Problems with having complex data hierarchy when modifying superclasses.
- Tightly coupling children class with ancestor one.
- Errors in hierarchy design makes it hard to refactor hierarchy.
- Keeping unnecessary data in children from ancestor.
- Breaking incapsulation with protected field access. Ability to break contract of ancestor class in child class.

Thinking in Object-Oriented way

Prefer composition over inheritance. Is-a versus has-a principle.

- Problems with having complex data hierarchy when modifying superclasses.
- Tightly coupling children class with ancestor one.
- Errors in hierarchy design makes it hard to refactor hierarchy.
- Keeping unnecessary data in children from ancestor.
- Breaking incapsulation with protected field access. Ability to break contract of ancestor class in child class.

Thinking in Object-Oriented way

Prefer composition over inheritance. Is-a versus has-a principle.

- Problems with having complex data hierarchy when modifying superclasses.
- Tightly coupling children class with ancestor one.
- Errors in hierarchy design makes it hard to refactor hierarchy.
- Keeping unnecessary data in children from ancestor.
- Breaking incapsulation with protected field access. Ability to break contract of ancestor class in child class.

Thinking in Object-Oriented way

Prefer composition over inheritance. Is-a versus has-a principle.

- Problems with having complex data hierarchy when modifying superclasses.
- Tightly coupling children class with ancestor one.
- Errors in hierarchy design makes it hard to refactor hierarchy.
- Keeping unnecessary data in children from ancestor.
- Breaking incapsulation with protected field access. Ability to break contract of ancestor class in child class.

Thinking in Object-Oriented way

Prefer composition over inheritance. Is-a versus has-a principle.

- Problems with having complex data hierarchy when modifying superclasses.
- Tightly coupling children class with ancestor one.
- Errors in hierarchy design makes it hard to refactor hierarchy.
- Keeping unnecessary data in children from ancestor.
- Breaking incapsulation with protected field access. Ability to break contract of ancestor class in child class.

Thinking in Object-Oriented way

Polymorphism

- Ad-hoc polymorphism, method overloading.
- Subtype polymorphism. Liskov substitution principle.
- Parametric polymorphism. Generics.

Thinking in Object-Oriented way

Polymorphism

- Ad-hoc polymorphism, method overloading.
- Subtype polymorphism. Liskov substitution principle.
- Parametric polymorphism. Generics.

Thinking in Object-Oriented way

Polymorphism

- Ad-hoc polymorphism, method overloading.
- Subtype polymorphism. Liskov substitution principle.
- Parametric polymorphism. Generics.

Thinking in Object-Oriented way

Abstract classes and interfaces

- Abstract class definition. Purpose. **Is-a** versus **Has-a**
- Interface definition. Contracts.
- No multiple inheritance allowed.

Thinking in Object-Oriented way

Abstract classes and interfaces

- Abstract class definition. Purpose. **Is-a** versus **Has-a**
- Interface definition. Contracts.
- No multiple inheritance allowed.

Thinking in Object-Oriented way

Abstract classes and interfaces

- Abstract class definition. Purpose. **Is-a** versus **Has-a**
- Interface definition. Contracts.
- No multiple inheritance allowed.

Thinking in Object-Oriented way

Design Patterns

- Program to interfaces - not implementations.
- Prefer composition over inheritance.
- Open-close principle.

Thinking in Object-Oriented way

Design Patterns

- Program to interfaces - not implementations.
- Prefer composition over inheritance.
- Open-close principle.

Thinking in Object-Oriented way

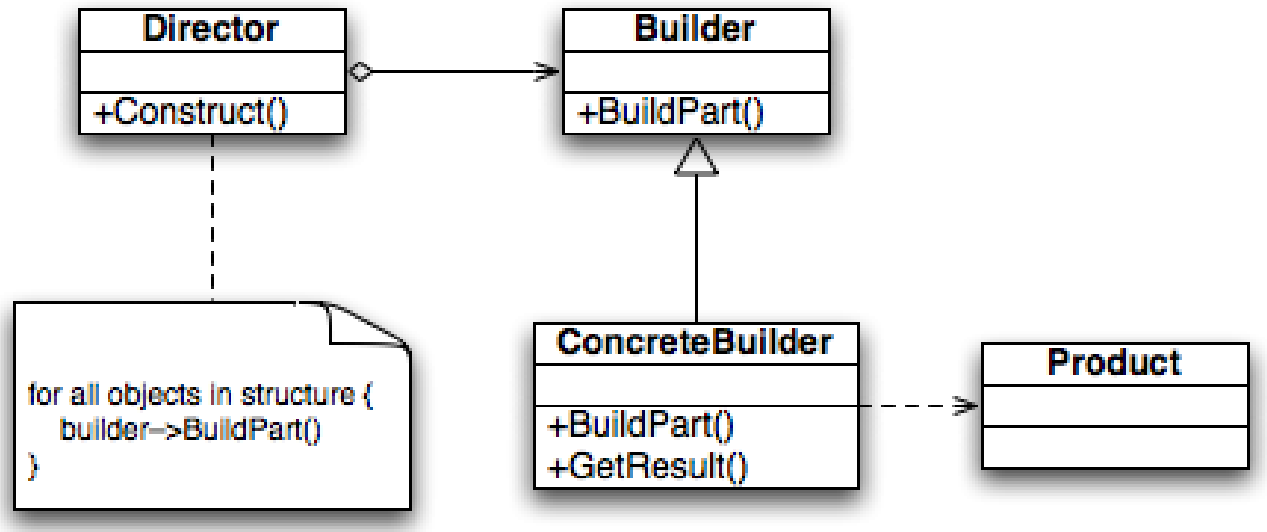
Design Patterns

- Program to interfaces - not implementations.
- Prefer composition over inheritance.
- Open-close principle.

Thinking in Object-Oriented way

Types of design patterns :: Creational patterns.

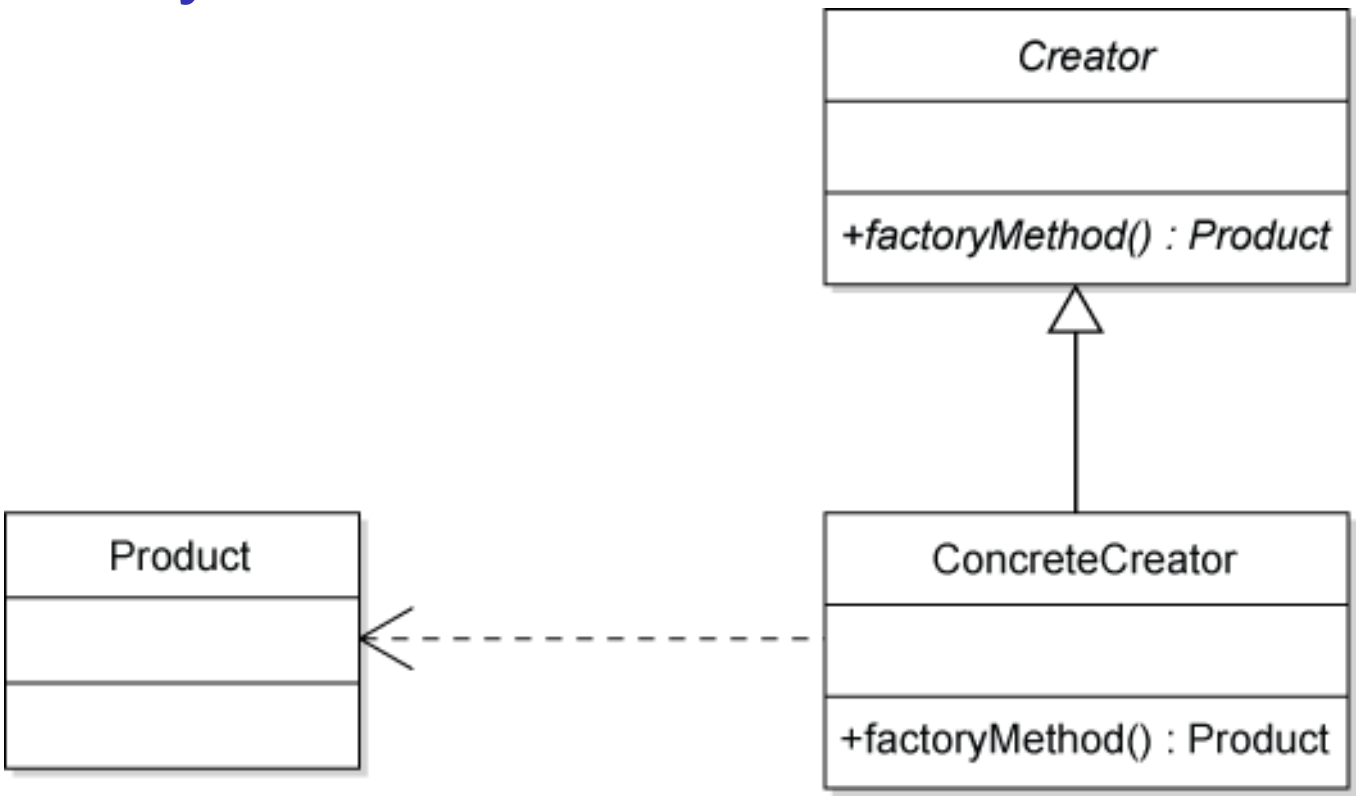
Builder



Thinking in Object-Oriented way

Types of design patterns :: Creational patterns.

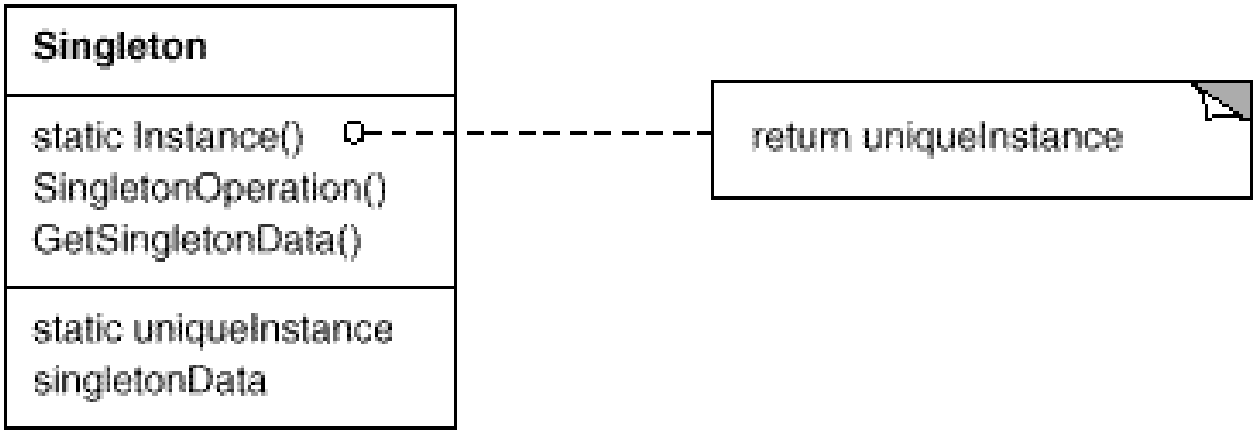
Factory method



Thinking in Object-Oriented way

Types of design patterns :: Creational patterns.

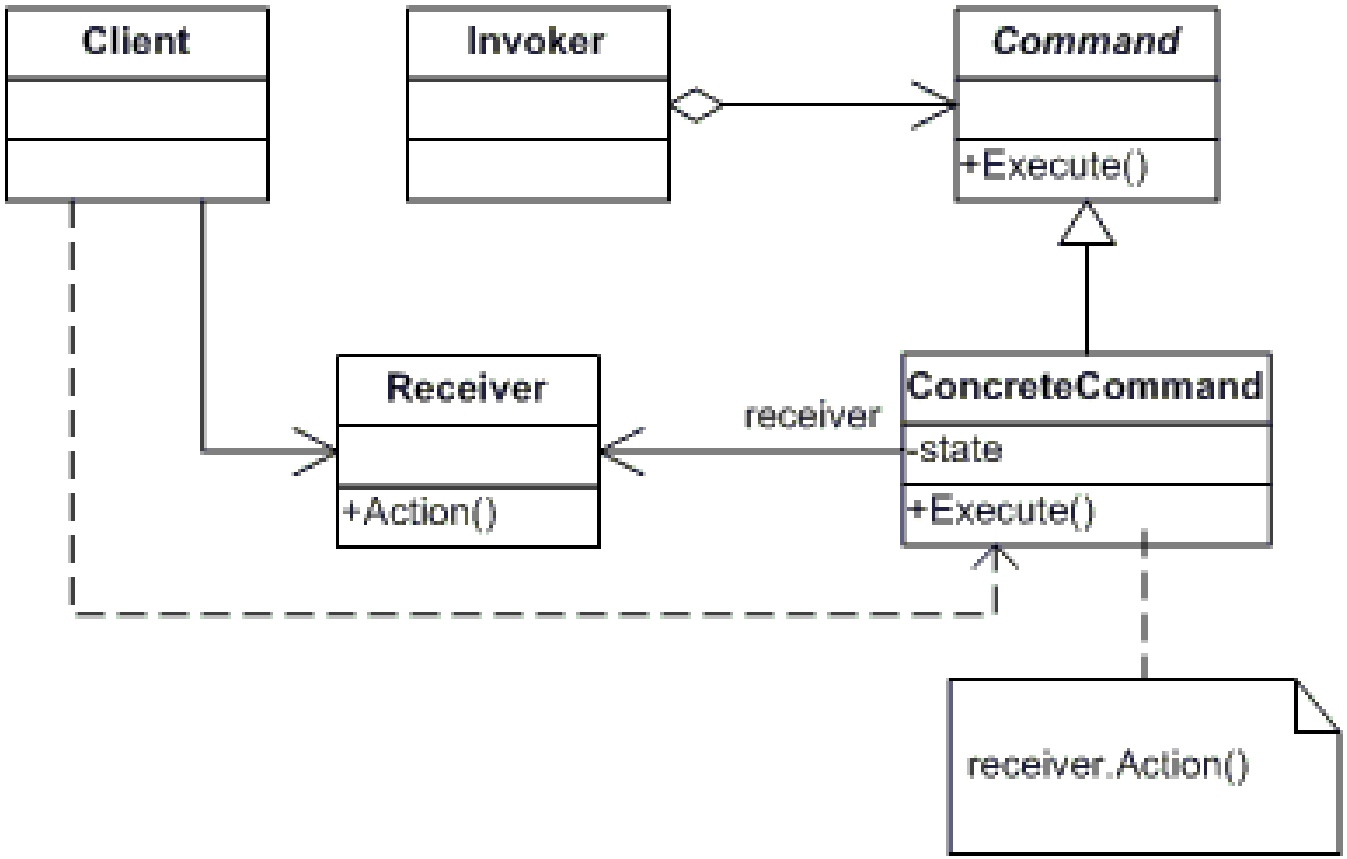
Singleton



Thinking in Object-Oriented way

Types of design patterns :: Behavioral patterns.

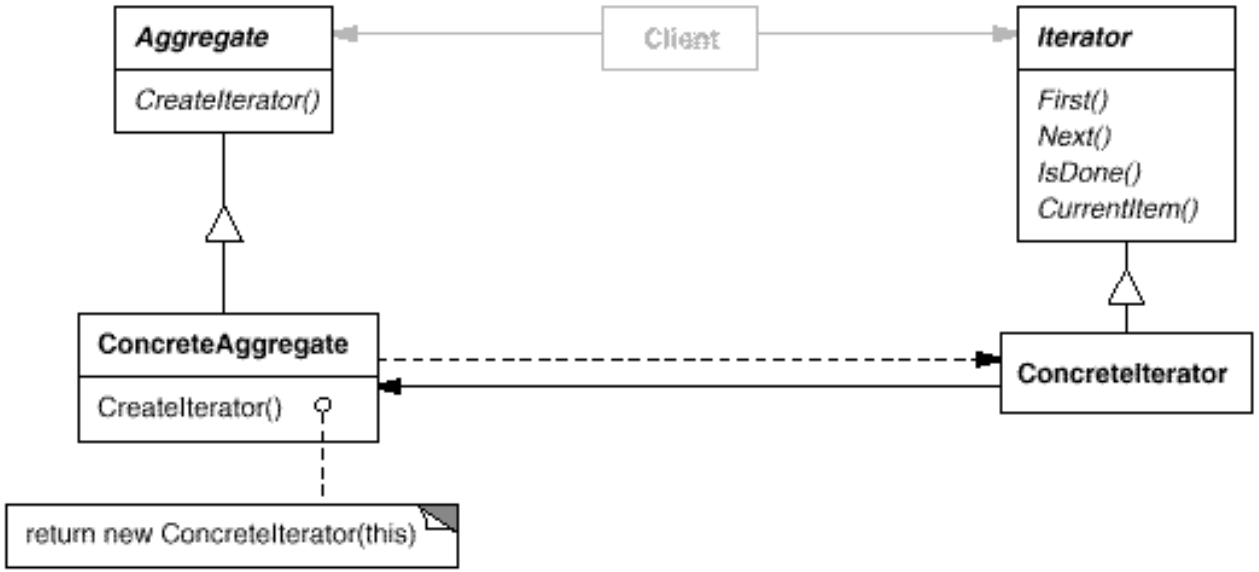
Command



Thinking in Object-Oriented way

Types of design patterns :: Behavioral patterns.

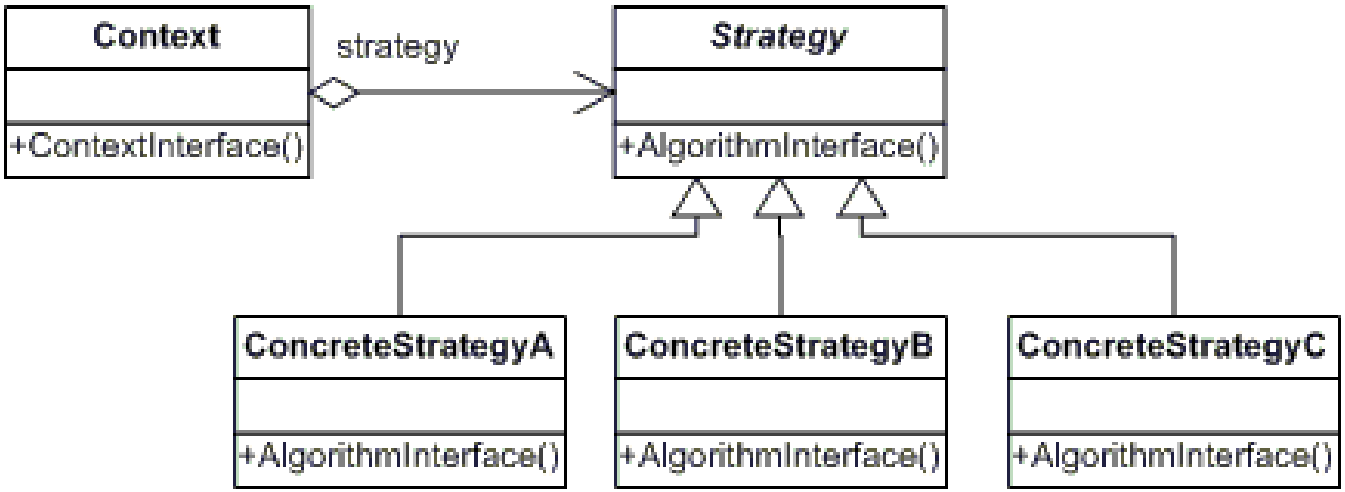
Iterator



Thinking in Object-Oriented way

Types of design patterns :: Behavioral patterns.

Strategy



Java and WEB applications.
HTTP Protocol.

HTTP protocol overview

- Request / response model
- Stateless
- Request headers
- Response headers
- Cookies
- Web sockets

Java and WEB applications.
HTTP Protocol.

HTTP protocol overview

- Request / response model
- Stateless
- Request headers
- Response headers
- Cookies
- Web sockets

Java and WEB applications.
HTTP Protocol.

HTTP protocol overview

- Request / response model
- Stateless
- Request headers
- Response headers
- Cookies
- Web sockets

Java and WEB applications.
HTTP Protocol.

HTTP protocol overview

- Request / response model
- Stateless
- Request headers
- Response headers
- Cookies
- Web sockets

Java and WEB applications.
HTTP Protocol.

HTTP protocol overview

- Request / response model
- Stateless
- Request headers
- Response headers
- Cookies
- Web sockets

Java and WEB applications.
HTTP Protocol.

HTTP protocol overview

- Request / response model
- Stateless
- Request headers
- Response headers
- Cookies
- Web sockets

Java and WEB applications.
HTML.

HTML overview

- HTML markup
- Hyperlinks
- Forms
- CSS
- JavaScript

Java and WEB applications.
HTML.

HTML overview

- HTML markup
- Hyperlinks
- Forms
- CSS
- JavaScript

Java and WEB applications.
HTML.

HTML overview

- HTML markup
- Hyperlinks
- Forms
- CSS
- JavaScript

Java and WEB applications.
HTML.

HTML overview

- HTML markup
- Hyperlinks
- Forms
- CSS
- JavaScript

Java and WEB applications.
HTML.

HTML overview

- HTML markup
- Hyperlinks
- Forms
- CSS
- JavaScript

Java and WEB applications.
Databases.

Data Definition Language

- Tables.
- Indexes. Unique indexes. Primary keys.
- Views.

Java and WEB applications.
Databases.

Data Definition Language

- Tables.
- Indexes. Unique indexes. Primary keys.
- Views.

Java and WEB applications.
Databases.

Data Definition Language

- Tables.
- Indexes. Unique indexes. Primary keys.
- Views.

Java and WEB applications.
JDBC.

Database connectivity

- JDBC overview.
- Database drivers.
- Connections.
- Statements and prepared statements.
- Result sets.

Java and WEB applications.
JDBC.

Database connectivity

- JDBC overview.
- Database drivers.
- Connections.
- Statements and prepared statements.
- Result sets.

Java and WEB applications.
JDBC.

Database connectivity

- JDBC overview.
- Database drivers.
- Connections.
- Statements and prepared statements.
- Result sets.

Java and WEB applications.
JDBC.

Database connectivity

- JDBC overview.
- Database drivers.
- Connections.
- Statements and prepared statements.
- Result sets.

Java and WEB applications.
JDBC.

Database connectivity

- JDBC overview.
- Database drivers.
- Connections.
- Statements and prepared statements.
- Result sets.

Java and WEB applications.
Servlets and Java Server Pages.

Java Servlets

- javax.servlet.http.HttpServlet
- javax.servlet.http.HttpServletRequest and javax.servlet.http.HttpServletResponse
- Sessions. Customer identification.
- web.xml definition.

Java and WEB applications.
Servlets and Java Server Pages.

Java Servlets

- javax.servlet.http.HttpServlet
- javax.servlet.http.HttpServletRequest and javax.servlet.http.HttpServletResponse
- Sessions. Customer identification.
- web.xml definition.

Java and WEB applications.
Servlets and Java Server Pages.

Java Servlets

- javax.servlet.http.HttpServlet
- javax.servlet.http.HttpServletRequest and javax.servlet.http.HttpServletResponse
- Sessions. Customer identification.
- web.xml definition.

Java and WEB applications.
Servlets and Java Server Pages.

Java Servlets

- javax.servlet.http.HttpServlet
- javax.servlet.http.HttpServletRequest and javax.servlet.http.HttpServletResponse
- Sessions. Customer identification.
- web.xml definition.

Java and WEB applications.
Servlets and Java Server Pages.

Java Server Pages

- JSP Model 1 and JSP Model 2. Scriplets.
- Model/View/Controller.
- Entry point.
- Request forwarding and including.

Java and WEB applications.
Servlets and Java Server Pages.

Java Server Pages

- JSP Model 1 and JSP Model 2. Scriplets.
- Model/View/Controller.
- Entry point.
- Request forwarding and including.

Java and WEB applications.
Servlets and Java Server Pages.

Java Server Pages

- JSP Model 1 and JSP Model 2. Scriplets.
- Model/View/Controller.
- Entry point.
- Request forwarding and including.

Java and WEB applications.
Servlets and Java Server Pages.

Java Server Pages

- JSP Model 1 and JSP Model 2. Scriplets.
- Model/View/Controller.
- Entry point.
- Request forwarding and including.

Java and WEB applications.
Servlet Containers.

Servlet containers

- Apache Tomcat
- Jetty

Java and WEB applications.
Servlet Containers.

Servlet containers

- Apache Tomcat
- Jetty

Software engineering

Real life development

- Metodologies (Waterfall, Agile, RUP, XP)
- Outsourcing. Bodyshops.
- Freelancing. Scriptlance, Elance, Odesk, Rentacoder.
- Shareware. Software directories. Digital river.

Software engineering

Real life development

- Metodologies (Waterfall, Agile, RUP, XP)
- Outsourcing. Bodyshops.
- Freelancing. Scriptlance, Elance, Odesk, Rentacoder.
- Shareware. Software directories. Digital river.

Software engineering

Real life development

- Metodologies (Waterfall, Agile, RUP, XP)
- Outsourcing. Bodyshops.
- Freelancing. Scriptlance, Elance, Odesk, Rentacoder.
- Shareware. Software directories. Digital river.

Software engineering

Real life development

- Metodologies (Waterfall, Agile, RUP, XP)
- Outsourcing. Bodyshops.
- Freelancing. Scriptlance, Elance, Odesk, Rentacoder.
- Shareware. Software directories. Digital river.