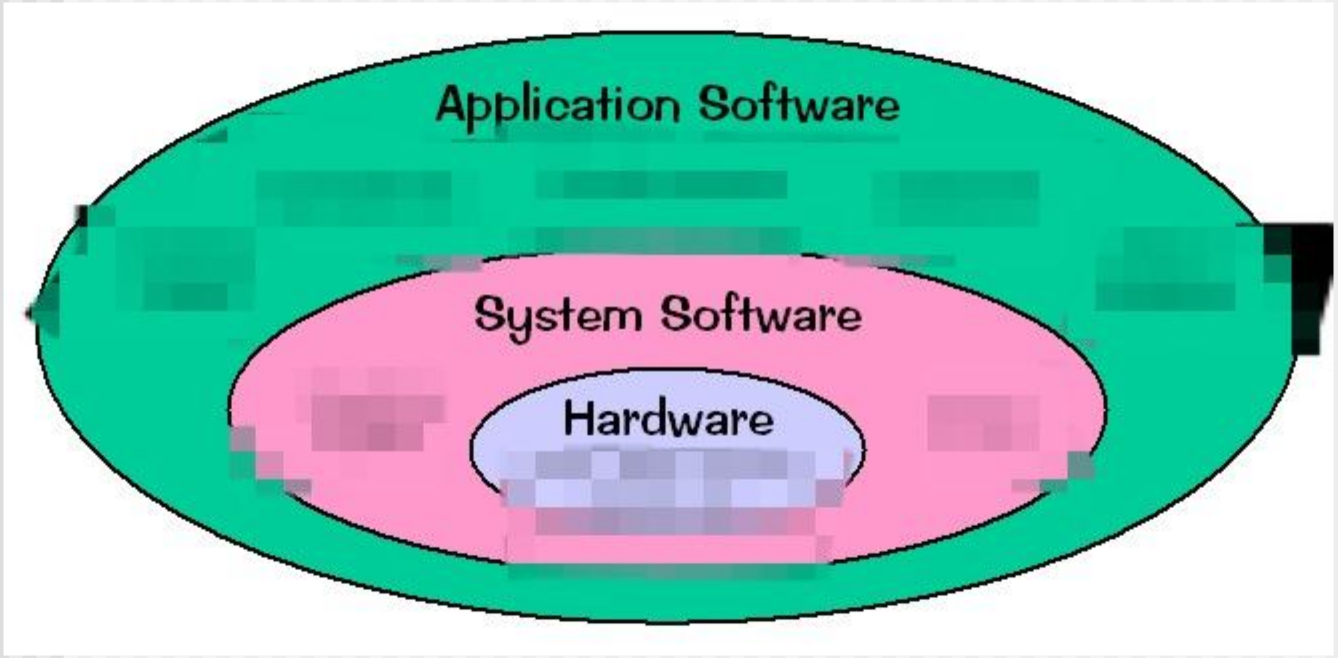
What are systems you build?



These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

Design & WebApps

"There are essentially two basic approaches to design: the artistic ideal of expressing yourself and the engineering ideal of solving a problem for a customer."

Jakob Nielsen

- *When should we emphasize WebApp design?*
  - when content and function are complex
  - when the size of the WebApp encompasses hundreds of content objects, functions, and analysis classes
  - when the success of the WebApp will have a direct impact on the success of the business

Design & WebApp Quality

- Security
  - Rebuff external attacks
  - Exclude unauthorized access
  - Ensure the privacy of users/customers
- Availability
  - the measure of the percentage of time that a WebApp is available for use
- Scalability
  - **Can** the WebApp and the systems with which it is interfaced handle significant variation in user or transaction volume
- Time to Market

Quality Dimensions for End-Users

- ***Time***
  - How much has a Web site changed since the last upgrade?
  - How do you highlight the parts that have changed?
- ***Structural***
  - How well do all of the parts of the Web site hold together.
  - Are all links inside and outside the Web site working?
  - Do all of the images work?
  - Are there parts of the Web site that are not connected?
- ***Content***
  - Does the content of critical pages match what is supposed to be there?
  - Do key phrases exist continually in highly-changeable pages?
  - Do critical pages maintain quality content from version to version?
  - What about dynamically generated HTML pages?

Quality Dimensions for End-Users

- ***Accuracy and Consistency***
  - Are today's copies of the pages downloaded the same as yesterday's? Close enough?
  - Is the data presented accurate enough? How do you know?
- ***Response Time and Latency***
  - Does the Web site server respond to a browser request within certain parameters?
  - In an E-commerce context, how is the end to end response time after a SUBMIT?
  - Are there parts of a site that are so slow the user declines to continue working on it?
- ***Performance***
  - Is the Browser-Web-Web site-Web-Browser connection quick enough?
  - How does the performance vary by time of day, by load and usage?
  - Is performance adequate for E-commerce applications?

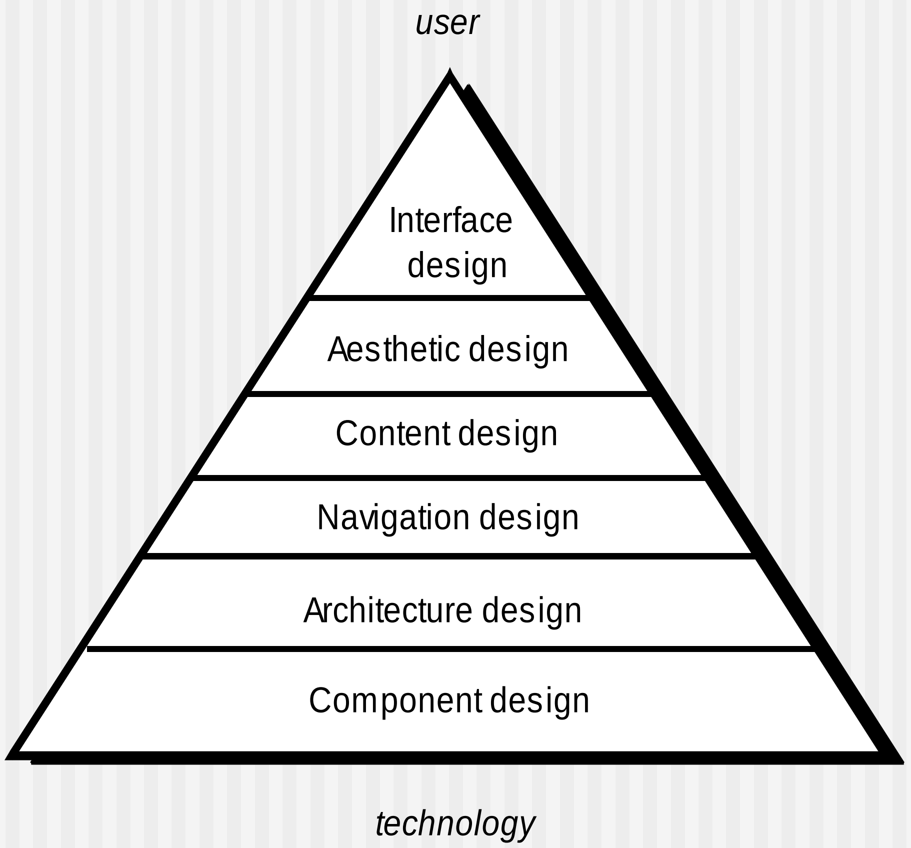
WebApp Design Goals

- Consistency
  - Content should be constructed consistently
  - Graphic design (aesthetics) should present a consistent look across all parts of the WebApp
  - Architectural design should establish templates that lead to a consistent hypermedia structure
  - Interface design should define consistent modes of interaction, navigation and content display
  - Navigation mechanisms should be used consistently across all WebApp elements

WebApp Design Goals

- Identity
  - Establish an “identity” that is appropriate for the business purpose
- Robustness
  - The user expects robust content and functions that are relevant to the user’s needs
- Navigability
  - designed in a manner that is intuitive and predictable
- Visual appeal
  - the look and feel of content, interface layout, color coordination, the balance of text, graphics and other media, navigation mechanisms must appeal to end-users
- Compatibility
  - With all appropriate environments and configurations

WebE Design Pyramid



WebApp Interface Design

- *Where am I?* The interface should
  - provide an indication of the WebApp that has been accessed
  - inform the user of her location in the content hierarchy.
- *What can I do now?* The interface should always help the user understand his current options
  - what functions are available?
  - what links are live?
  - what content is relevant?
- *Where have I been, where am I going?* The interface must facilitate navigation.
  - Provide a "map" (implemented in a way that is easy to understand) of where the user has been and what paths may be taken to move elsewhere within the WebApp.

Effective WebApp Interfaces

- Bruce Tognozzi [TOG01] suggests…
  - Effective interfaces are visually apparent and forgiving, instilling in their users a sense of control. Users quickly see the breadth of their options, grasp how to achieve their goals, and do their work.
  - Effective interfaces do not concern the user with the inner workings of the system. Work is carefully and continuously saved, with full option for the user to undo any activity at any time.
  - Effective applications and services perform a maximum of work, while requiring a minimum of information from users.

Interface Design Principles-I

- Anticipation—A WebApp should be designed so that it anticipates the use's next move.
- Communication—The interface should communicate the status of any activity initiated by the user
- Consistency—The use of navigation controls, menus, icons, and aesthetics (e.g., color, shape, layout)
- Controlled autonomy—The interface should facilitate user movement throughout the WebApp, but it should do so in a manner that enforces navigation conventions that have been established for the application.
- Efficiency—The design of the WebApp and its interface should optimize the user's work efficiency, not the efficiency of the Web engineer who designs and builds it or the client-server environment that executes it.

Interface Design Principles-II

- Focus—The WebApp interface (and the content it presents) should stay focused on the user task(s) at hand.
- Fitt's Law—"The time to acquire a target is a function of the distance to and size of the target."
- Human interface objects—A vast library of reusable human interface objects has been developed for WebApps.
- Latency reduction—The WebApp should use multi-tasking in a way that lets the user proceed with work as if the operation has been completed.
- Learnability— A WebApp interface should be designed to minimize learning time, and once learned, to minimize relearning required when the WebApp is revisited.

Interface Design Principles-III

- Maintain work product integrity—A work product (e.g., a form completed by the user, a user specified list) must be automatically saved so that it will not be lost if an error occurs.
- Readability—All information presented through the interface should be readable by young and old.
- Track state—When appropriate, the state of the user interaction should be tracked and stored so that a user can logoff and return later to pick up where she left off.
- Visible navigation—A well-designed WebApp interface provides "the illusion that users are in the same place, with the work brought to them."

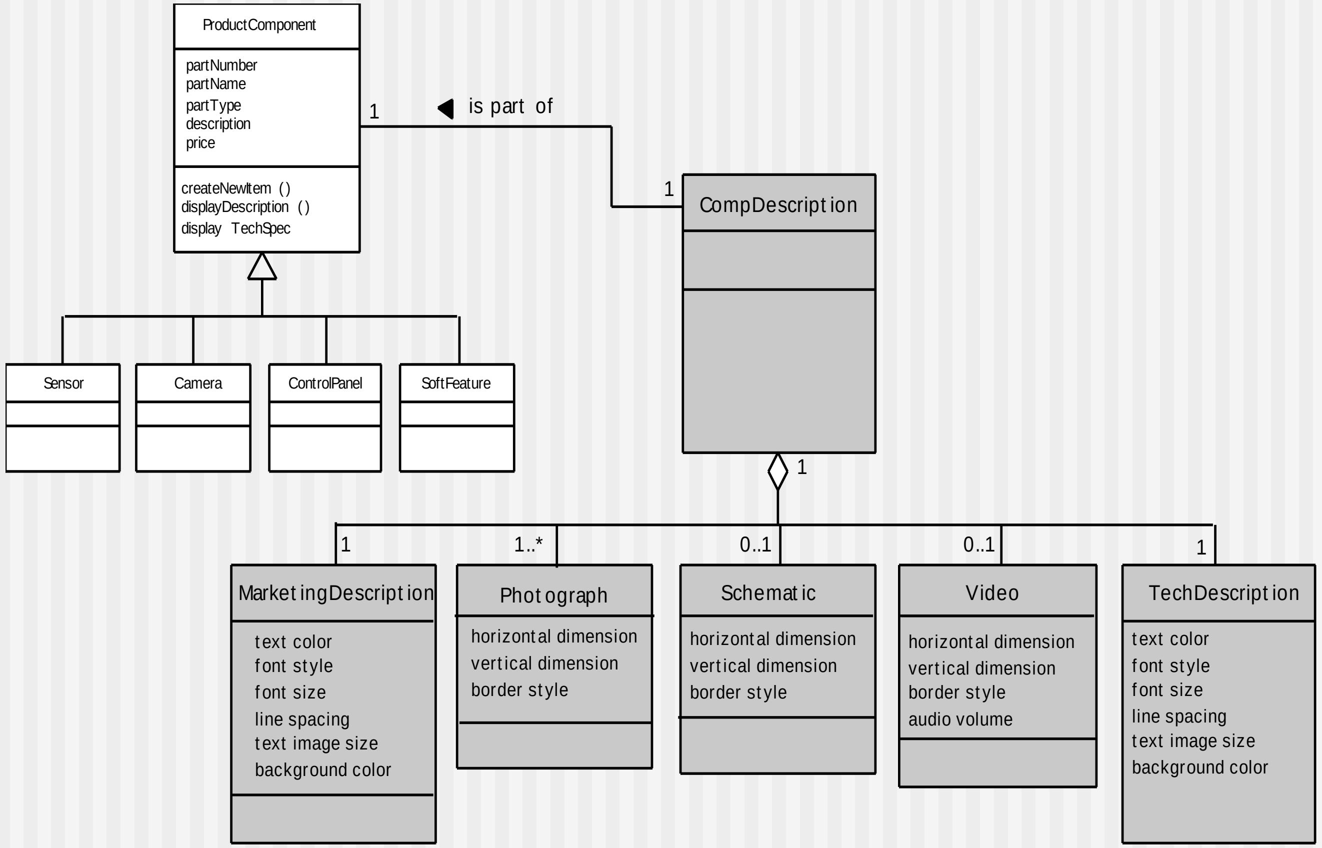
Aesthetic Design

- Don’t be afraid of white space.
- Emphasize content.
- Organize layout elements from top-left to bottom right.
- Group navigation, content, and function geographically within the page.
- Don’t extend your real estate with the scrolling bar.
- Consider resolution and browser window size when designing layout.

Content Design

- Develops a design representation for content objects
  - For WebApps, a content object is more closely aligned with a data object for conventional software
- Represents the mechanisms required to instantiate their relationships to one another.
  - analogous to the relationship between analysis classes and design components described in Chapter 11
- A content object has attributes that include content-specific information and implementation-specific attributes that are specified as part of design

Design of Content Objects



Architecture Design

- *Content architecture* focuses on the manner in which content objects (or composite objects such as Web pages) are structured for presentation and navigation.
  - The term information architecture is also used to connote structures that lead to better organization, labeling, navigation, and searching of content objects.
- *WebApp architecture* addresses the manner in which the application is structured to manage user interaction, handle internal processing tasks, effect navigation, and present content.
- Architecture design is conducted in parallel with interface design, aesthetic design and content design.

Design Patterns vs. Frameworks

Even though they are two very different things, one can argue they **both solve a software architecture problem**

- a design pattern solves many software architecture issues (about creation, behavior, concurrency, ...) with different pre-defined *design*. (design being an implementation of an architecture topic)
- a framework is based on the **Hollywood Principle** ("Don't call us, we call you"), where you implement some high level requirements as specified, and leave the framework do the rest of the work, calling your implementations.
- A key difference is the **scope cohesion**:
- **design pattern have a *tight* scope**:
  - class design patterns (involves classes)
  - business design patterns (involves business workflows)
  - application design patterns (involves applications)
- **framework has a *large* scope**:
  For instance, **.NET is a framework** composed of:
  - a language (C#)
  - a runtime environment (CLR)
  - a collection of libraries
    Just develop what you need and let the .Net framework call your classes.

Design Patterns

- Each of us has encountered a design problem and silently thought: *I wonder if anyone has developed a solution to for this?*
  - What if there was a standard way of describing a problem (so you could look it up), and an organized method for representing the solution to the problem?
- *Design patterns* are a codified method for describing problems and their solution allows the software engineering community to capture design knowledge in a way that enables it to be reused.

Design Patterns

- *Each pattern describes a problem that occurs over and over again in our environment and then describes the core of the solution to that problem in such a way that you can use the solution a million times over without ever doing it the same way twice.*
    - Christopher Alexander, 1977
- “a three-part rule which expresses a relation between a certain context, a problem, and a solution.”

Basic Concepts

- *Context* allows the reader to understand the environment in which the problem resides and what solution might be appropriate within that environment.
- A set of requirements, including limitations and constraints, acts as a *system of forces* that influences how
  - the problem can be interpreted within its context and
  - how the solution can be effectively applied.

Effective Patterns

- Coplien [Cop05] characterizes an effective design pattern in the following way:
  - *It solves a problem*: Patterns capture solutions, not just abstract principles or strategies.
  - *It is a proven concept*: Patterns capture solutions with a track record, not theories or speculation.
  - *The solution isn't obvious*: Many problem-solving techniques (such as software design paradigms or methods) try to derive solutions from first principles. The best patterns *generate* a solution to a problem indirectly--a necessary approach for the most difficult problems of design.
  - *It describes a relationship*: Patterns don't just describe modules, but describe deeper system structures and mechanisms.
  - *The pattern has a significant human component (minimize human intervention).* All software serves human comfort or quality of life; the best patterns explicitly appeal to aesthetics and utility.

Generative Patterns

- *Generative patterns* describe an important and repeatable aspect of a system and then provide us with a way to build that aspect within a system of forces that are unique to a given context.
- A collection of generative design patterns could be used to “generate” an application or computer-based system whose architecture enables it to adapt to change.

Kinds of Patterns

- *Architectural patterns* describe broad-based design problems that are solved using a structural approach.
- *Data patterns* describe recurring data-oriented problems and the data modeling solutions that can be used to solve them.
- *Component patterns* (also referred to as *design patterns*) address problems associated with the development of subsystems and components, the manner in which they communicate with one another, and their placement within a larger architecture
- *Interface design patterns* describe common user interface problems and their solution with a system of forces that includes the specific characteristics of end-users.
- *WebApp patterns* address a problem set that is encountered when building WebApps and often incorporates many of the other patterns categories just mentioned.

Kinds of Patterns

- ***Creational patterns*** focus on the “creation, composition, and representation of objects, e.g.,
  - **Abstract factory pattern:** centralize decision of what factory to instantiate
  - **Factory method pattern:** centralize creation of an object of a specific type choosing one of several implementations
- ***Structural patterns*** focus on problems and solutions associated with how classes and objects are organized and integrated to build a larger structure, e.g.,
  - **Adapter pattern:** 'adapts' one interface for a class into one that a client expects
  - **Aggregate pattern:** a version of the Composite pattern with methods for aggregation of children
- ***Behavioral patterns*** address problems associated with the assignment of responsibility between objects and the manner in which communication is effected between objects, e.g.,
  - **Chain of responsibility pattern:** Command objects are handled or passed on to other objects by logic-containing processing objects
  - **Command pattern:** Command objects encapsulate an action and its parameters

Thinking in Patterns

- Shalloway and Trott [Sha05] suggest the following approach that enables a designer to think in patterns:
  - 1. Be sure you understand the big picture—the context in which the software to be built resides. The requirements model should communicate this to you.
  - 2. Examining the big picture, extract the patterns that are present at that level of abstraction.
  - 3. Begin your design with 'big picture' patterns that establish a context or skeleton for further design work.
  - 4. "Work inward from the context" [Sha05] looking for patterns at lower levels of abstraction that contribute to the design solution.
  - 5. Repeat steps 1 to 4 until the complete design is fleshed out.
  - 6. Refine the design by adapting each pattern to the specifics of the software you're trying to build.

Common Design Mistakes

- Not enough time has been spent to understand the underlying problem, its context and forces, and as a consequence, you select a pattern that looks right, but is inappropriate for the solution required.
- Once the wrong pattern is selected, you refuse to see your error and force fit the pattern.
- In other cases, the problem has forces that are not considered by the pattern you’ve chosen, resulting in a poor or erroneous fit.
- Sometimes a pattern is applied too literally and the required adaptations for your problem space are not implemented.

Outline

- **Introduction**
  - What is Framework
  - What is WAF
- Java Web Aplication Framework
- Conclusion

What is Framework?

- Introduction
  - **What is Framework**
  - What is WAF
- Java Web Aplication Framework
- Conclusion

What is Framework?

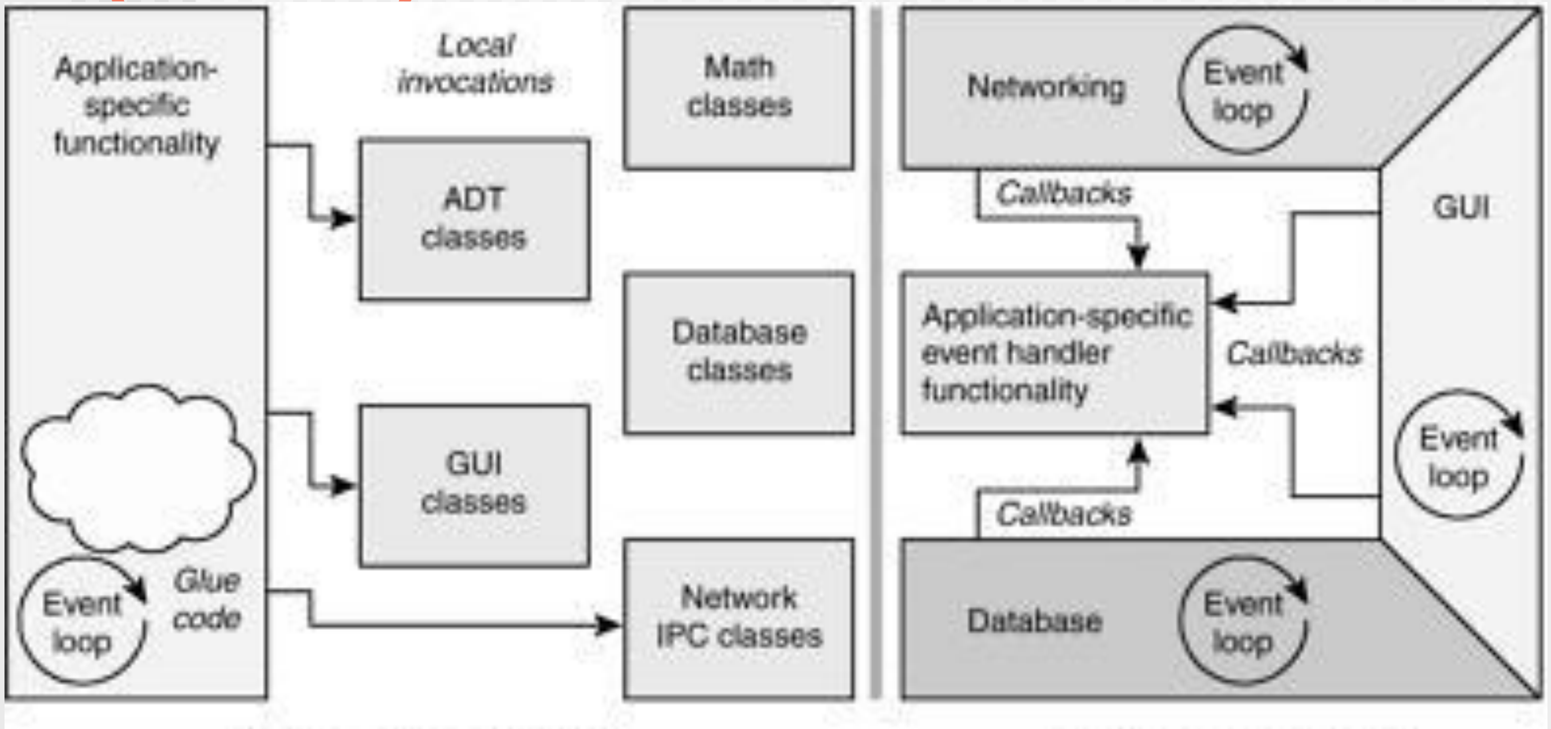
- In information systems environment, a framework is a defined support structure in which other software applications can be organized and developed.
- A software framework is a reusable design and building blocks for a software system and/or subsystem

What is Framework? (cont.)

- In an object-oriented environment, a framework consists of abstract and concrete classes. Instantiation of such a framework consists of composing and subclassing the existing classes

- Software frameworks rely on the Hollywood Principle: "Don’t call us, we’ll call you."

What is Framework? (cont.)



(1) Class library architecture (2) Framework architecture

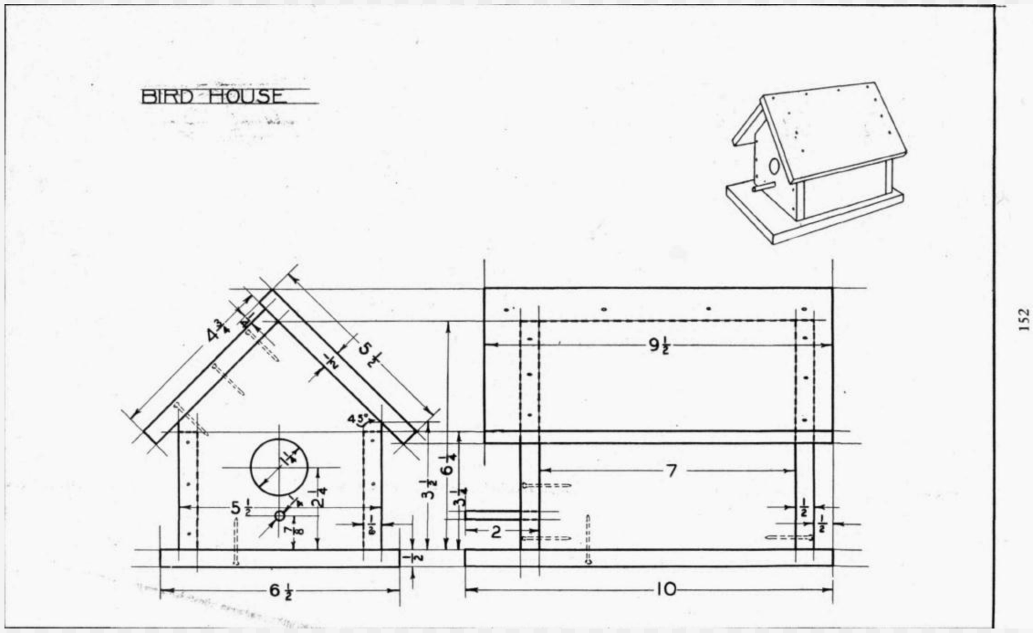
Why we use Framework?

- The software frameworks significantly reduce the amount of time, effort, and resources required to develop and maintain applications.

What is Framework?

- Introduction
  - What is Framework
  - **What is WAF**
- Java Web Aplication Framework
- Conclusion

What is the difference between architecture and framework?



Technical Motivation

Software

- becoming more pervasive
- increasing in size and complexity
- expected to provide product flexibility

Therefore, need to increase productivity, manage complexity, and provide flexibility to systems builders

Competitive Motivation

“Heads down” programming expertise has become commodity skill available globally

Therefore, our students need to

- become skilled domain analysts
- learn to adapt software to local needs
- master higher level concepts and skills

One Response: Focus on Software Families

Software family is set of programs

- Sharing many common properties
- Worthwhile to study as a group before each individually
- Known as software frameworks or product lines

David Parnas:

“If you are developing a family of programs, you must do that consciously, or you will incur unnecessary long-term costs.”

How Much Progress?

- In 1976, Parnas published the seminal article “On the Design and Development of Program Families.”

- In 2001, Parnas wrote:

  “Although there is now growing … interest and some evidence of … success in applying the idea, the majority of industrial programmers seem to ignore it in their rush to produce code.”

Foundational Concepts

- Classic concepts (Parnas)
  - information hiding modules
  - abstract interfaces
- Object-oriented programming
  - inheritance
  - polymorphism
- Design patterns

Information-Hiding Modules

Module

- Work assignment given to a programmer or group of programmers
- Good characteristics: independent development, comprehensible, changeable

Information hiding

- Design modules around assumptions likely to change
- Hide a design decision within a module
    - as its **secret**

Abstract Interfaces

Interface

- Set of assumptions programmer must make about another module to demonstrate correctness of his module
  - not just syntax, also semantics

Abstract interface

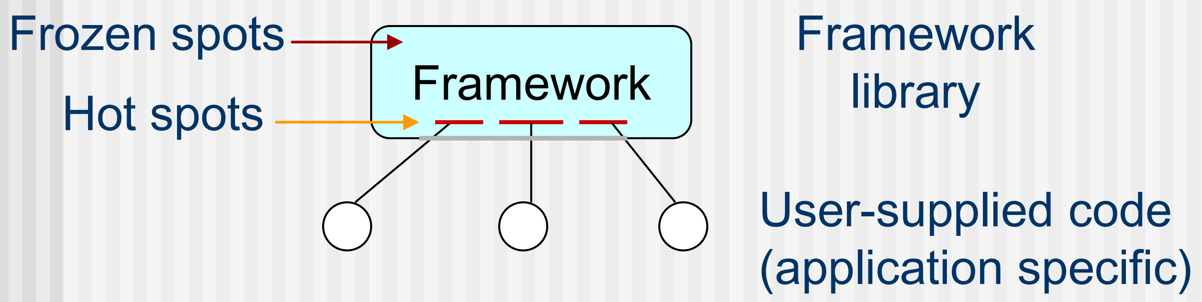
- Module interface that does not change when one module implementation is substituted for another

Design Patterns

- Well-established solutions to recurring design problems

Software Framework

- Generic application allowing creation of members of family of related programs
- Reusable design expressed as set of abstract classes and way they collaborate
- Common and variable aspects known as *frozen spots* and *hot spots*



Hot Spot Subsystem

- *Template method* defines a common behavior for family
  - part of frozen spot implementation
- Template method calls *hook methods* to invoke behaviors specific to family member
  - hook methods form **hot spot subsystem**
  - set of hook methods form abstract interface to information hiding module

Example:
Divide and Conquer Algorithms

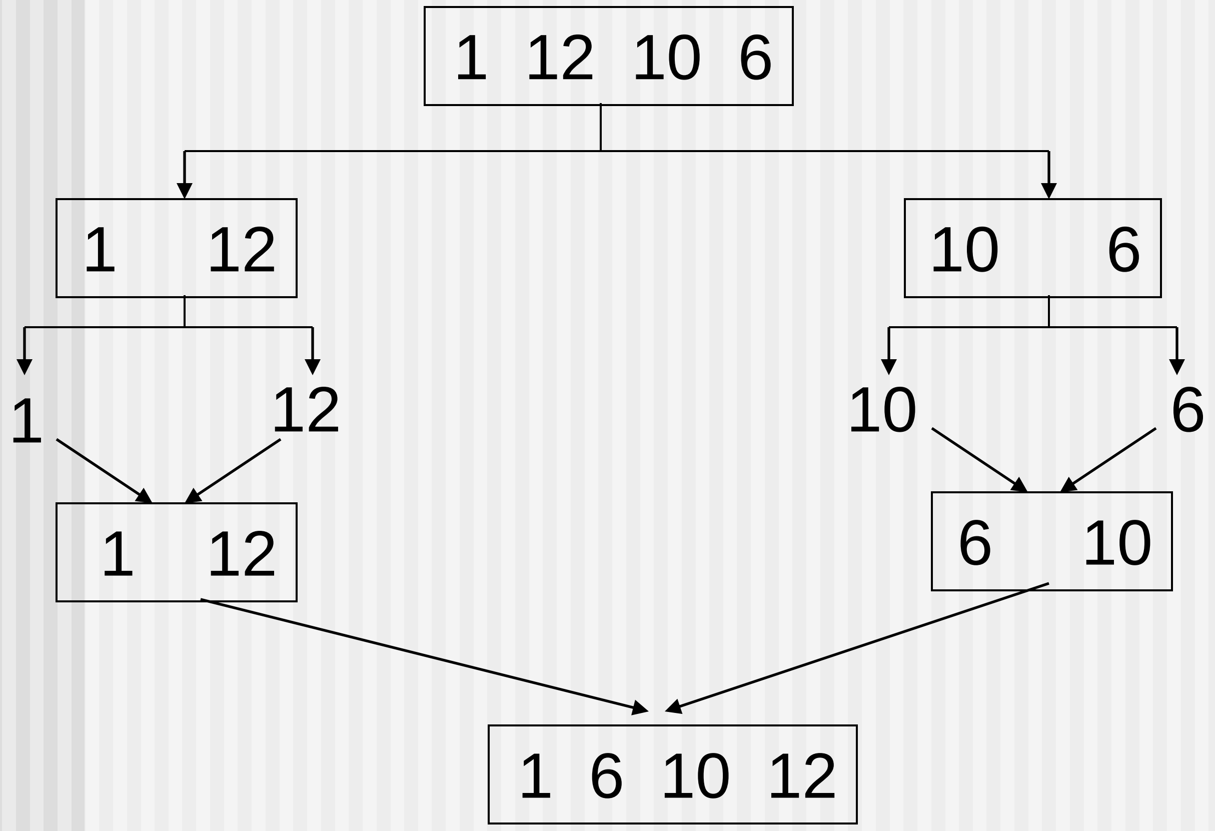
- Divide original problem into subproblems of same type
- Solve each subproblem independently
- Combine subproblem solutions into solution for original problem

What are some examples?

Well-Known Examples

- Mergesort
- Quicksort
- Heapify
- Binary search
- Fast matrix multiplication
- Fast Fourier Transform (FFT)

Example: Mergesort



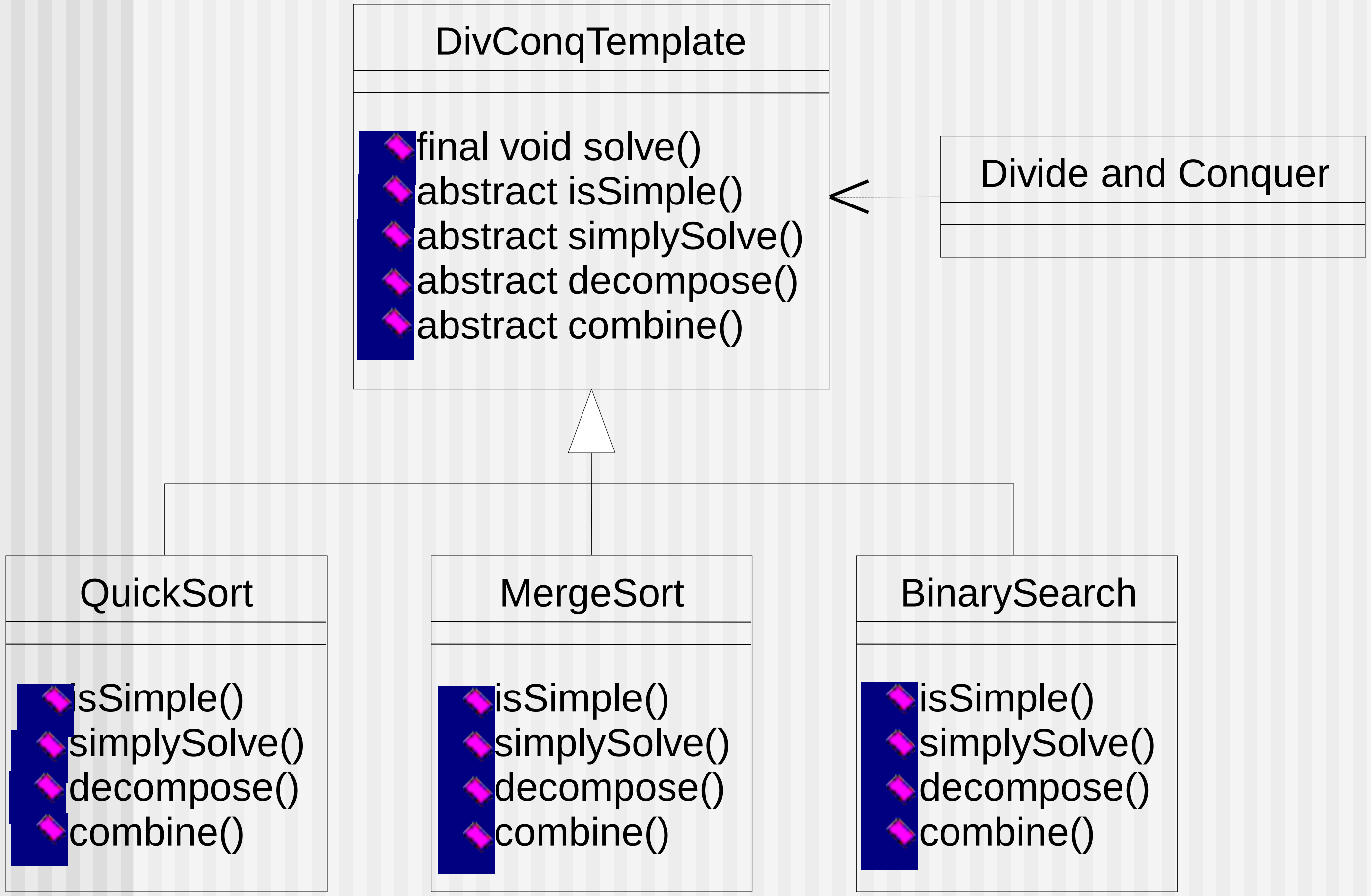
Divide and Conquer Pseudocode

```
function solve(p)
{ if isSimple(p)
     return simplySolve(p);
  else
  {
     Problem [] sp = decompose(p);
     for (i = 0; i<sp.length; i = i+1)
        sol[i] = solve(sp[i]);
        return combine (sol);
  }
}
```

Divide & Conquer Functions

- template method:
  - `solve()`
- hook methods:
  - `isSimple()`
  - `simplySolve()`
  - `decompose()`
  - `combine()`
- other: problem and solution descriptions

Framework: Template Method Pattern



Template Method Class

```
abstract public class DivConqTemplate
{ final public Solution solve (Problem p)
  { Problem[] pp;
    if (isSimple(p)) { return simplySolve(p); }
    else             { pp = decompose(p);     }
    Solution[] ss = new Solution[pp.length];
    for(int i=0; i < pp.length; i++)
    { ss[i] = solve(pp[i]); }
    return combine(p,ss);
  }
  abstract protected boolean isSimple (Problem p);
  abstract protected Solution simplySolve (Problem p);
  abstract protected Problem[] decompose (Problem p);
  abstract protected Solution
    combine (Problem p, Solution[] ss);
}
```

Framework Application: Quicksort

- Divide and conquer task: sort a sequence in place
- Mapping problem into divide and conquer framework
  - `decompose`: partition into two segments about a pivot value (rearranges values in array)
  - recursively sort each segment until just one element (`isSimple` and `simplySolve`)
  - `combine`: values already in needed location
- Describing the problem and solution
  - identify sequence of values
  - identify beginning and ending elements of segment

Problem and Solution Description

Simplifying assumption: sort integer arrays

- `Problem` description for sort
  - overall array
  - beginning and ending indices of segment
- `Solution` description for sort
  - same as `Problem` description
  - except array values rearranged in place
- `QuickSortDesc` to represent both

QuickSortDesc Class

```
public class QuickSortDesc
    implements Problem, Solution
{ public QuickSortDesc
    (int[] arr, int first, int last)
  { this.arr = arr;
    this.first = first; this.last = last;
  }
  public int getFirst () { return first; }
  public int getLast ()  { return last;  }
  public int[] getArr () { return arr;   }
  private int[] arr;
  private int first, last;
}
```

Quicksort Subclass (1 of 3)

```
public class QuickSort extends DivConqTemplate
{ protected boolean isSimple (Problem p)
  { return (
      ((QuickSortDesc)p).getFirst()
          >=
      ((QuickSortDesc)p).getLast() );
  }

  protected Solution simplySolve (Problem p)
  { return (Solution) p ; }
```

Quicksort Subclass (2 of 3)

```
protected Problem[] decompose (Problem p)
{ int first = ((QuickSortDesc)p).getFirst();
  int last  = ((QuickSortDesc)p).getLast();
  int[] a   = ((QuickSortDesc)p).getArr ();
  int x     = a[first];  // pivot value
  int sp    = first;
  for (int i = first + 1; i <= last; i++)
  { if (a[i] < x)  { swap (a, ++sp, i); } }
  swap (a, first, sp);
  Problem[] ps = new QuickSortDesc[2];
  ps[0] = new QuickSortDesc(a,first,sp-1);
  ps[1] = new QuickSortDesc(a,sp+1,last);
  return ps;
}
```

Quicksort Subclass (3 of 3)

```
  protected Solution combine
       (Problem p, Solution[] ss)
  { return (Solution) p; } // should make sure adj

  private void swap (int [] a, int first, int last)
  { int temp = a[first];
    a[first] = a[last];
    a[last]  = temp;
  }
}
```

To Build a Framework

- Represent *common aspects* as abstract classes and concrete template methods
- Represent *variable aspects* as abstract hook methods and some concrete hook methods in hot spot subsystems
- Organize using unification or separation principle

To Use a Framework

- Implement concrete hook methods
- Plug subsystems into hot spots

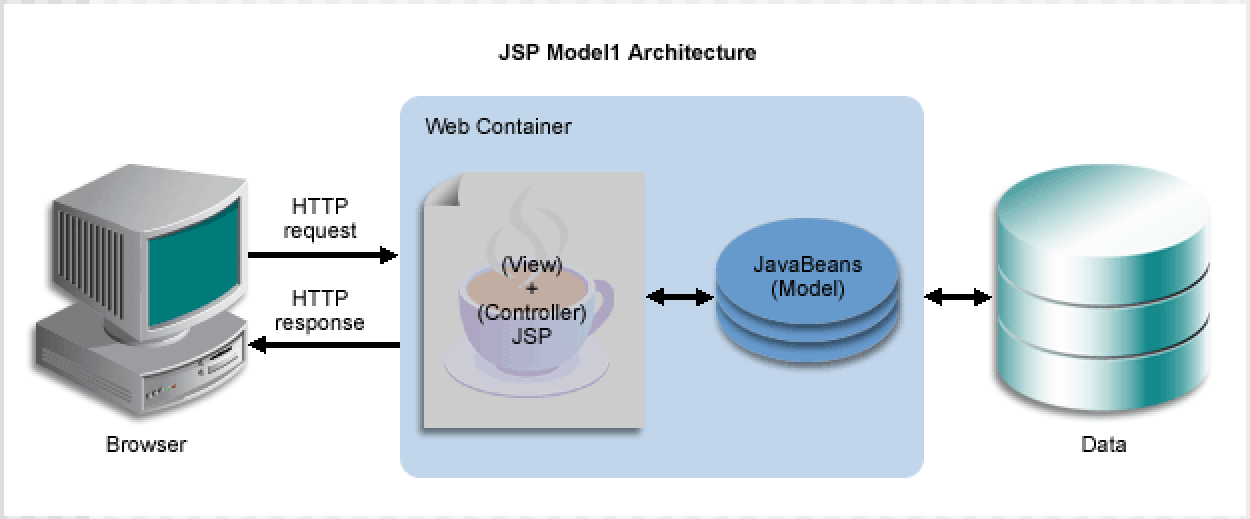
What is WAF?

- A Web Application Framework (WAF) is a reusable, skeletal, semi-complete modular platform that can be specialized to produce custom web applications , which commonly serve the web browsers via the Http's protocol.

- WAF usually implements the Model-View-Controller (MVC) design pattern, typically in the **Model 2** architecture to develop request-response web-based applications on the Java EE and .Net models.

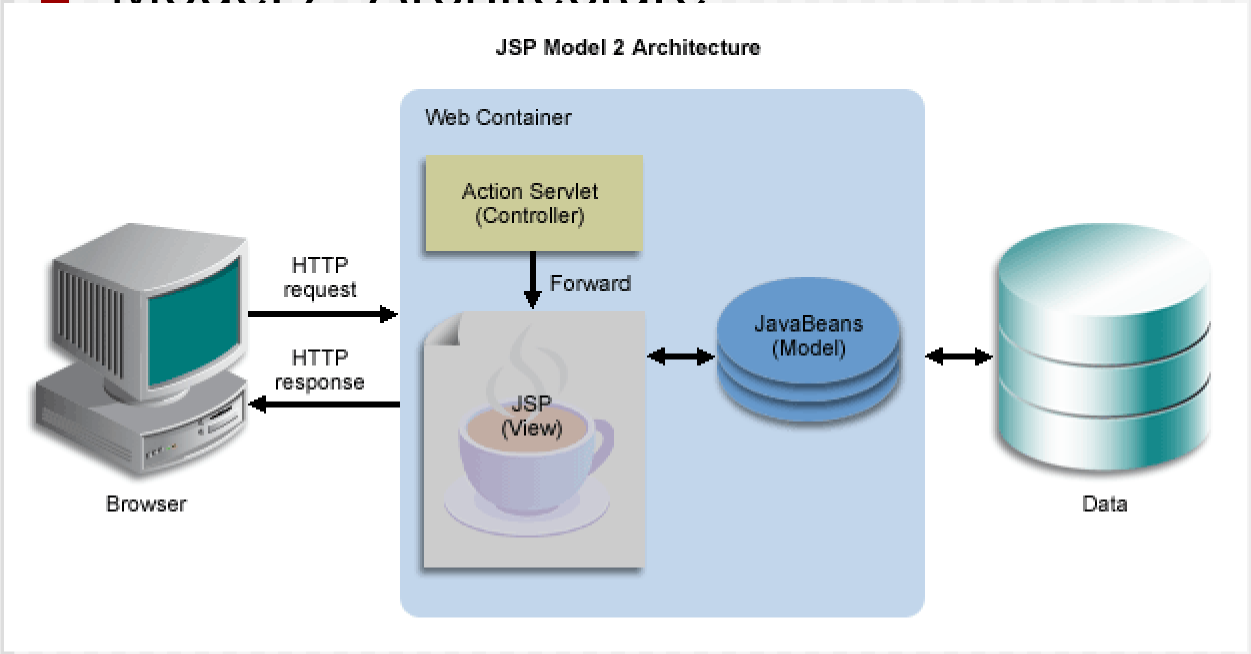
What is WAF? (cont.)

- Model 1 Architecture

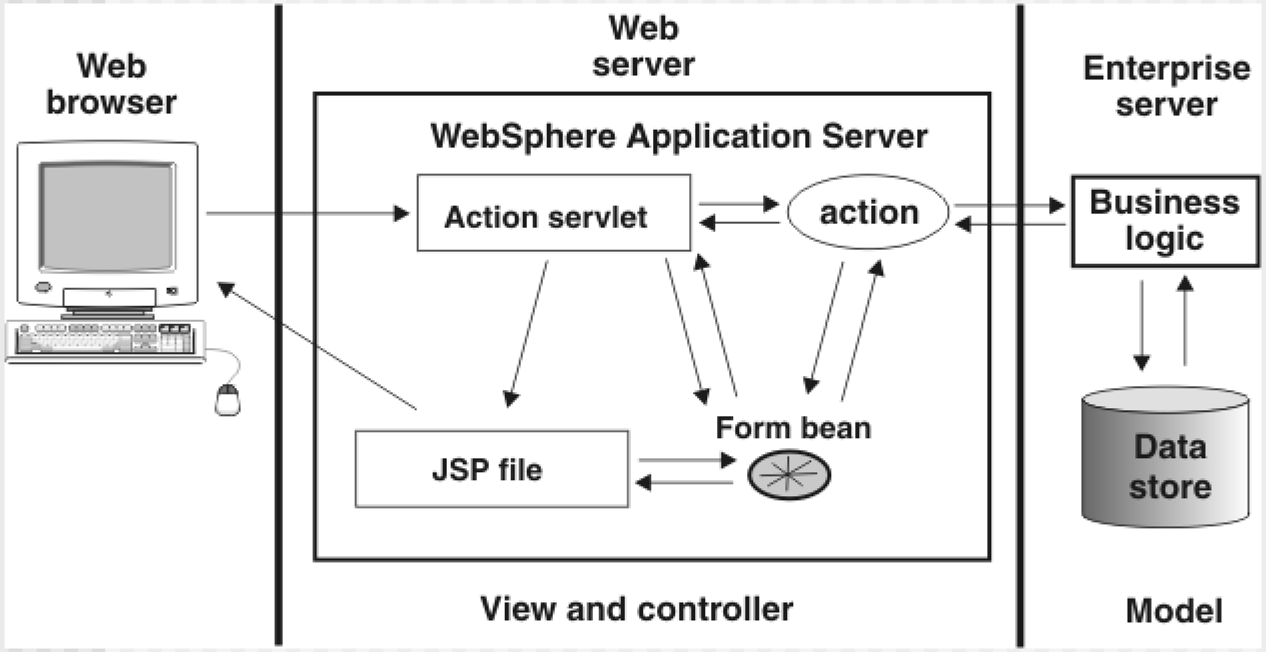


What is WAF? (cont.)

- Model 2 Architecture



Model-View-Controller



Why we use WAF?

- Virtually all web applications have a common set of basic requirements, such as **user management** e.g., **secure user login**, **password recovery**), group management, and access authorization.
- A Web Application Framework usually includes all these functionalities, refined through hundreds of production deployments, freeing developers to focus on the needs of their specific application.

Why we use WAF?

- WAFs store important data in a relational database and they interact with users via a web-based user interface.
- Any application written on top of a Web Application Framework can transparently and immediately take advantage of these basic services.

- Introduction
- Java Web Aplication Framework
- Conclusion

Java Web Aplication Framework

- Introduction
- **Java Web Aplication Framework**
  - Request-based Framework
  - Component-based Framework
  - Hybrid – Meta Framework
  - RIA-based Framework
- Conclusion

Request-based Framework

- Introduction
- Java Web Aplication Framework
  - **Request-based Framework**
  - Component-based Framework
  - Hybrid – Meta Framework
  - RIA-based Framework
- Conclusion

Request-based Framework

- A *Request-based Framework* is very close to the original CGI specification.
- It uses controllers and actions that directly handle incoming requests.
- Each request is essentially stateless.

Request-based Framework

- Introduction
- Java Web Aplication Framework
  - Request-based Framework
    - Struts
    - WebWork
    - Beehive , Stripes
  - Component-based Framework
  - Hybrid – Meta Framework
  - RIA-based Framework
- Conclusion

Request-based Framework

- Introduction
- Java Web Aplication Framework
  - Request-based Framework
    - **Struts**
    - WebWork
    - Beehive , Stripes
  - Component-based Framework
  - Hybrid – Meta Framework
  - RIA-based Framework
- Conclusion

Struts

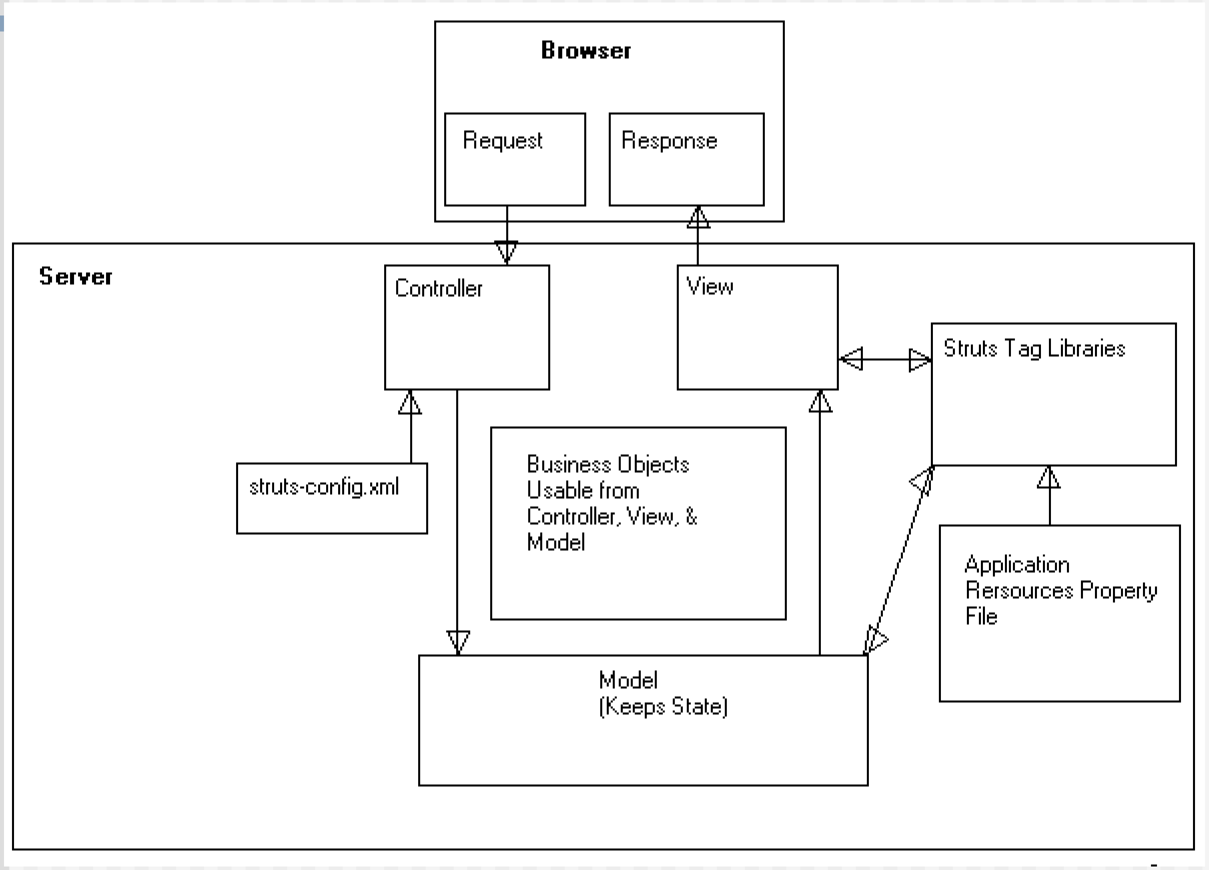
- Struts was originally developed by Craig McClanahan and donated to the Apache Foundation in May 2000.

- Struts has been a de facto framework with a strong and vibrant user community.

Struts (cont.)

- Struts uses and extends the Java Servlet API to adopt the "Model 2" approach, a variation of the classic Model-View-Controller (MVC) design pattern.

Struts (cont.) MVC Design pattern



Struts Tag Libraries

- HTML Tags
- Bean Tags
- Logic Tags
- Template Tags
- Custom Tags

Struts HTML Tags

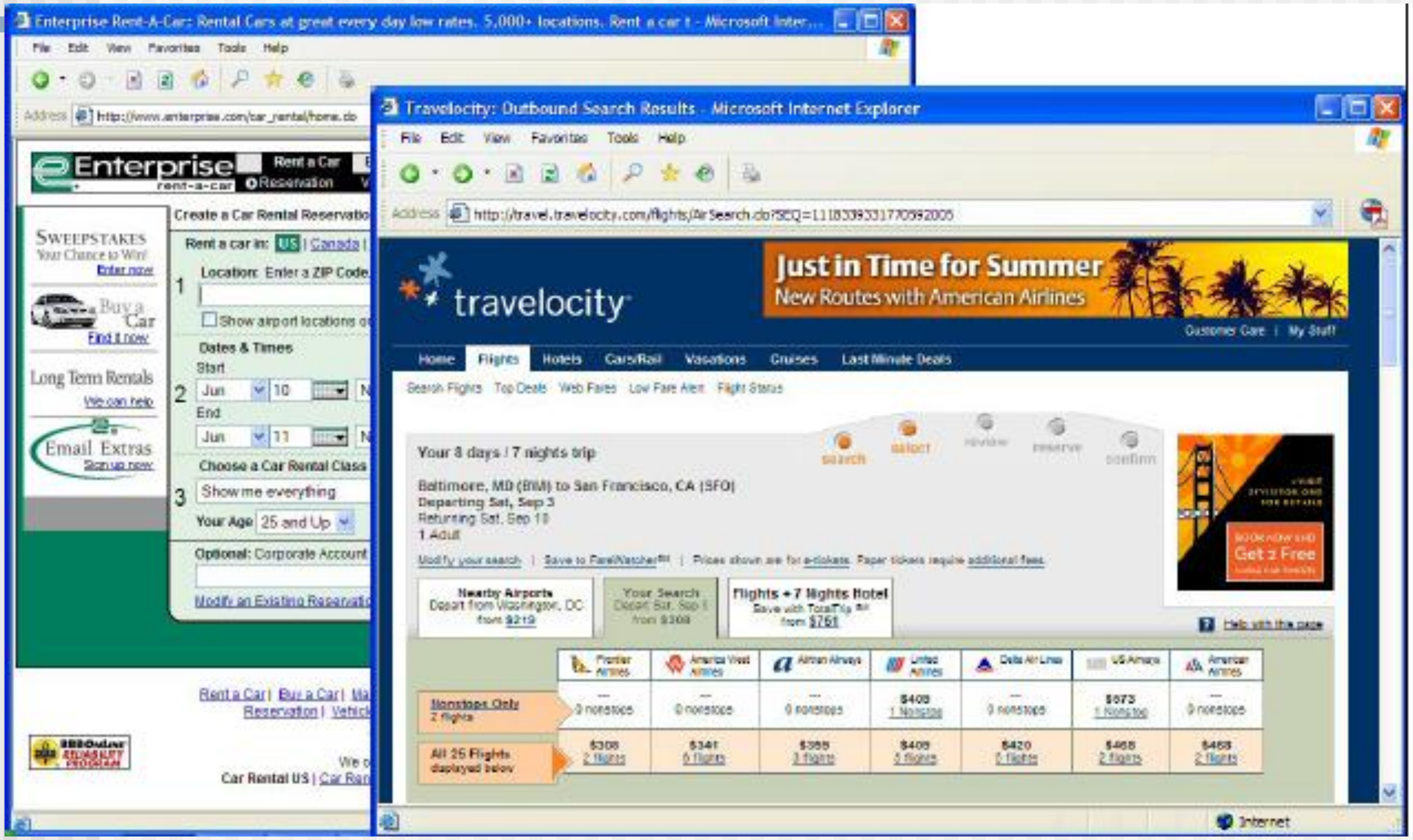
- The tags in the Struts HTML library form a bridge between a JSP view and the other components of a Web application.
- Since a dynamic Web application often depends on gathering data from a user, input forms play an important role in the Struts framework

Application Set-up

- Jar files
- Tag libraries
- Struts-config dtd
- Application.properties
- Apache vhost file
- Web.xml
- Struts-config.xml

Project Refinery, Inc.

Some Struts Sites: Travel



Enterprise Rent-A-Car: Rental Cars at great every day low rates. 5,000+ locations. Rent a car t - Microsoft Inter...

http://www.enterprise.com/car_rental/home.do

Enterprise rent-a-car

Rent a Car | Reservation

Create a Car Rental Reservation

Rent a car in: US | Canada |

1 Location: Enter a ZIP Code

Show airport locations

2 Dates & Times

Start: Jun 10

End: Jun 11

3 Choose a Car Rental Class: Show me everything

Your Age: 25 and Up

Optional: Corporate Account

Modify an Existing Reservation

SWEEPSTAKES Your Chance to Win! Enter now

Buy a Car Find it now

Long Term Rentals We can help

Email Extras Sign up now

Rent a Car | Buy a Car | Reservation | Vehicle

Car Rental US | Car Rental

Travelocity: Outbound Search Results - Microsoft Internet Explorer

http://travel.travelocity.com/flights/AirSearch.do?SEQ=1118339331770592005

travelocity

Just in Time for Summer — New Routes with American Airlines

Customer Care | My Stuff

Home | Flights | Hotels | Cars/Rail | Vacations | Cruises | Last Minute Deals

Search Flights | Top Deals | Web Fares | Low Fare Alert | Flight Status

search · select · review · reserve · confirm

Your 8 days / 7 nights trip

Baltimore, MD (BWI) to San Francisco, CA (SFO)
Departing Sat, Sep 3
Returning Sat, Sep 10
1 Adult

Modify your search | Save to FareWatcher | Prices shown are for e-tickets. Paper tickets require additional fees.

Nearby Airports — Depart from Washington, DC from $219

Your Search — Depart Sat, Sep 3 from $308

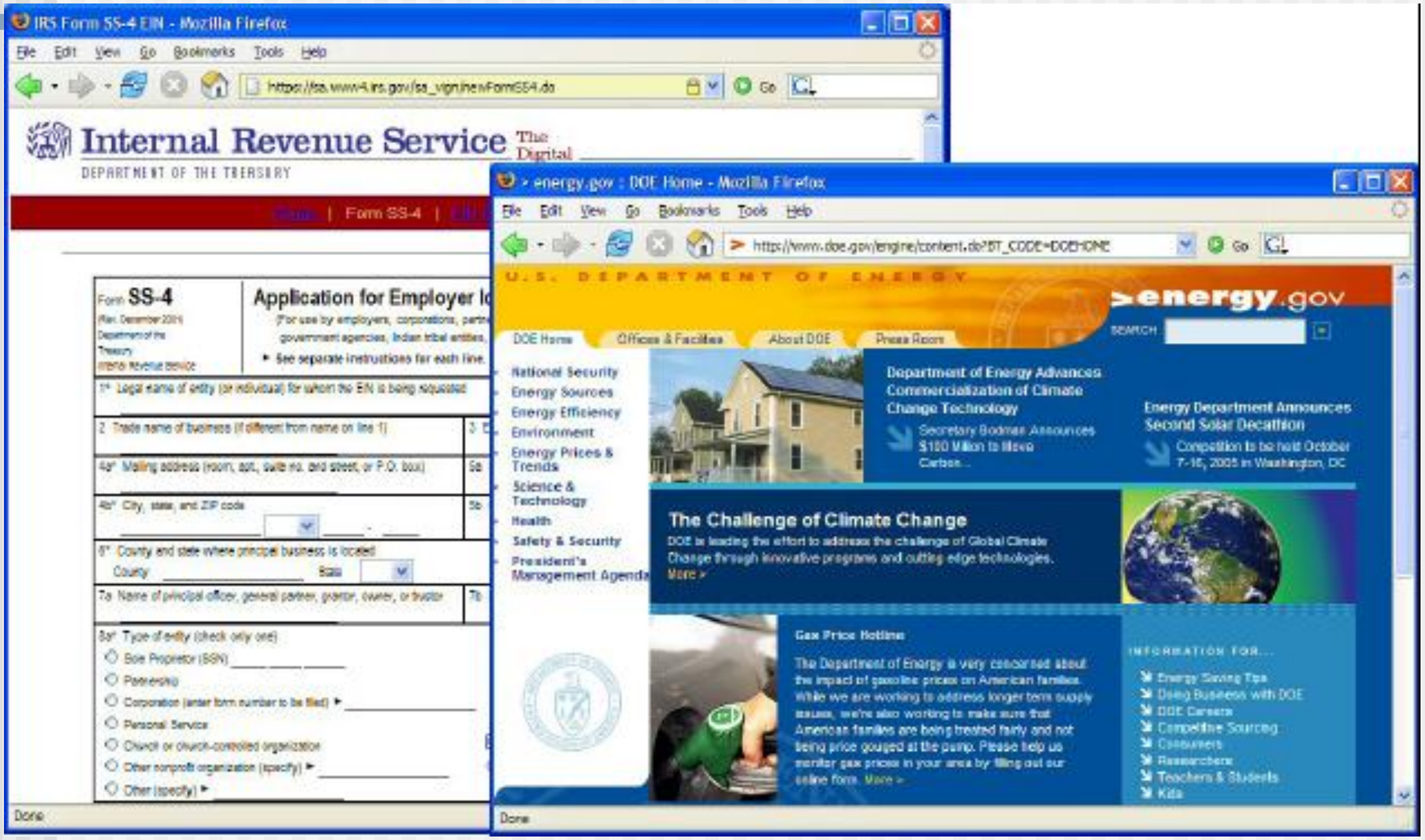
Flights + 7 Nights Hotel — Save with TotalTrip from $761

Help with this page

	Frontier Airlines	America West Airlines	AirTran Airways	United Airlines	Delta Air Lines	US Airways	American Airlines
Nonstops Only 2 flights	— 0 nonstops	— 0 nonstops	— 0 nonstops	$408 1 Nonstop	— 0 nonstops	$673 1 Nonstop	— 0 nonstops
All 25 Flights displayed below	$308 2 flights	$341 6 flights	$388 3 flights	$408 5 flights	$420 5 flights	$468 2 flights	$468 2 flights

Internet

Some Struts Sites: Government



Form SS-4 (Rev. December 2001) Department of the Treasury Internal Revenue Service

Application for Employer Id

(For use by employers, corporations, partn government agencies, Indian tribal entities,

▸ See separate instructions for each line.

1* Legal name of entity (or individual) for whom the EIN is being requested

2 Trade name of business (if different from name on line 1)

4a* Mailing address (room, apt., suite no. and street, or P.O. box)

4b* City, state, and ZIP code

6* County and state where principal business is located
County State

7a Name of principal officer, general partner, grantor, owner, or trustor

8a* Type of entity (check only one)
- Sole Proprietor (SSN)
- Partnership
- Corporation (enter form number to be filed) ▸
- Personal Service
- Church or church-controlled organization
- Other nonprofit organization (specify) ▸
- Other (specify) ▸

Department of Energy Advances Commercialization of Climate Change Technology
Secretary Bodman Announces $100 Million to Move Carbon...

Energy Department Announces Second Solar Decathlon
Competition to be held October 7-16, 2005 in Washington, DC

The Challenge of Climate Change
DOE is leading the effort to address the challenge of Global Climate Change through innovative programs and cutting edge technologies. More >

Gas Price Hotline
The Department of Energy is very concerned about the impact of gasoline prices on American families. While we are working to address longer term supply issues, we're also working to make sure that American families are being treated fairly and not being price gouged at the pump. Please help us monitor gas prices in your area by filling out our online form. More >

WebWork

- Introduction
- Java Web Aplication Framework
  - Request-based Framework
    - Struts
    - **WebWork**
    - Beehive , Stripes
  - Component-based Framework
  - Hybrid – Meta Framework
  - RIA-based Framework
- Conclusion

WebWork

- WebWork was originally developed by Rickard Oberg in 2001, and released as an open source project on SourceForge in March 2002.
- WebWork provides robust support for building reusable UI templates, such as form controls, UI themes, internationalization, dynamic form parameter mapping to JavaBeans, and robust client and server side validation.

Component-based Framework

- Introduction
- Java Web Aplication Framework
  - Request-based Framework
  - **Component-based Framework**
  - Hybrid – Meta Framework
  - RIA-based Framework
- Conclusion

Component-based Framework

- A *Component-based Framework* abstracts the internals of the request handling and encapsulates the logic into reusable components, often independent from the web medium
- The state is automatically handled by the framework, based on the data that is present in each component instance.
- Together with some form of event handling, this development model is very similar to the features offered by desktop GUI toolkits.

Component-based Framework

- Introduction
- Java Web Aplication Framework
  - Request-based Framework
  - Component-based Framework
    - JSF
    - Tapestry
    - Wicket
  - Hybrid – Meta Framework
  - RIA-based Framework
- Conclusion

Component-based Framework

- Introduction
- Java Web Aplication Framework
  - Request-based Framework
  - Component-based Framework
    - **Jave Server Faces (JSF)**
    - Tapestry
    - Wicket
  - Hybrid – Meta Framework
  - RIA-based Framework
- Conclusion

Jave Server Faces (JSF)

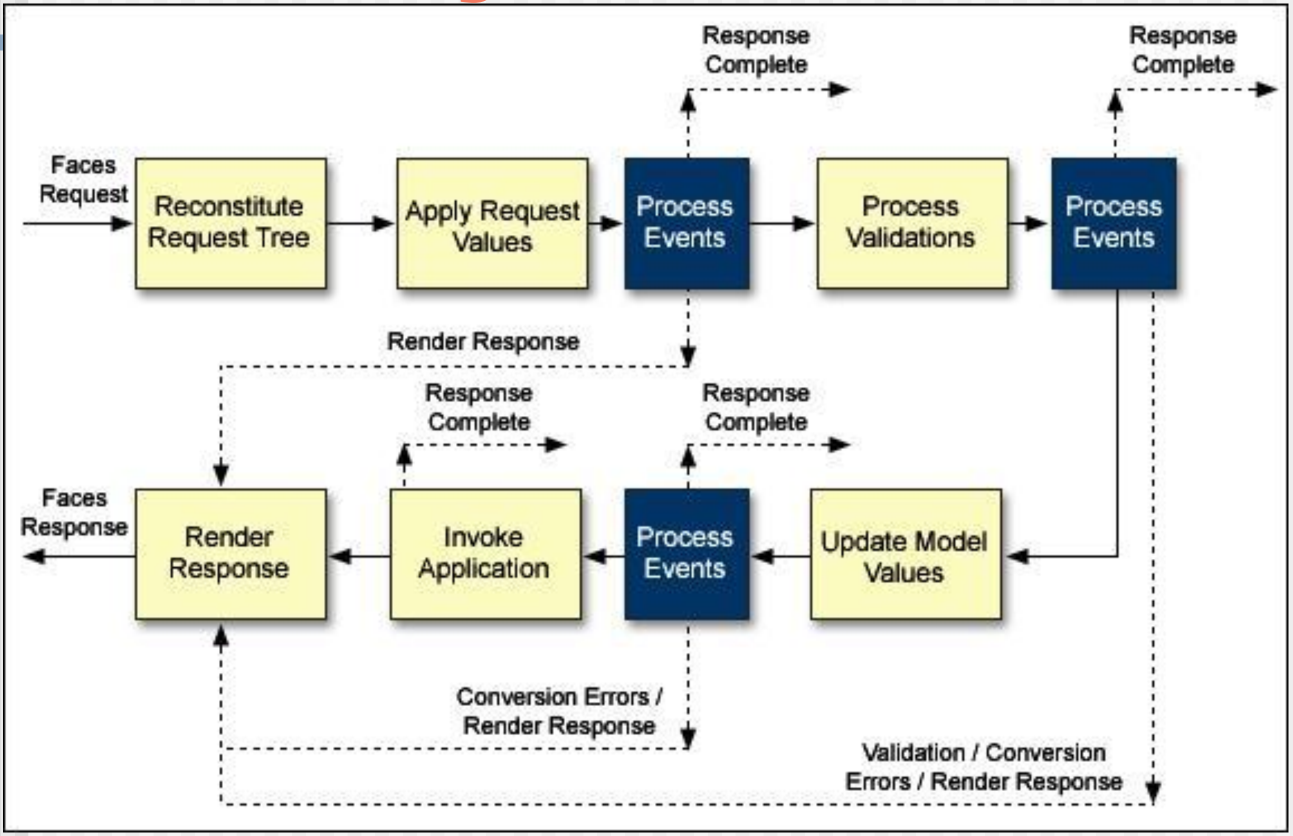
- JavaServer Faces (JSF) is a server-side user interface component framework for Java-based Web applications.

- JSF contains an API for representing UI components and **managing their state**; **handling events**, **server-side validation**, and **data conversion**; defining **page navigation**; supporting **internationalization** and **accessibility**; and providing **extensibility** for all these features

JSF Solutions

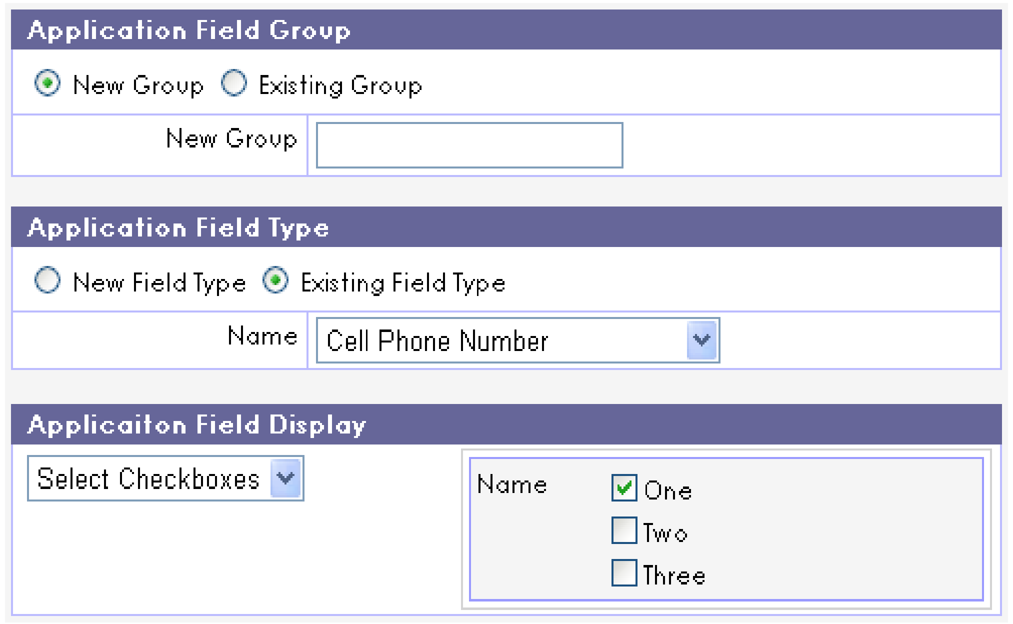
- UI components
- State management
- Event handling
- Input validation
- Page navigation
- Internationalization and accessibility.
- Custom tag library

JSF Life Cycle



UI Components (Standard)

Some of the standard JavaServer Faces Components



Application Field Group

[● New Group] ○ Existing Group

New Group

Application Field Type

○ New Field Type [● Existing Field Type]

Name: Cell Phone Number

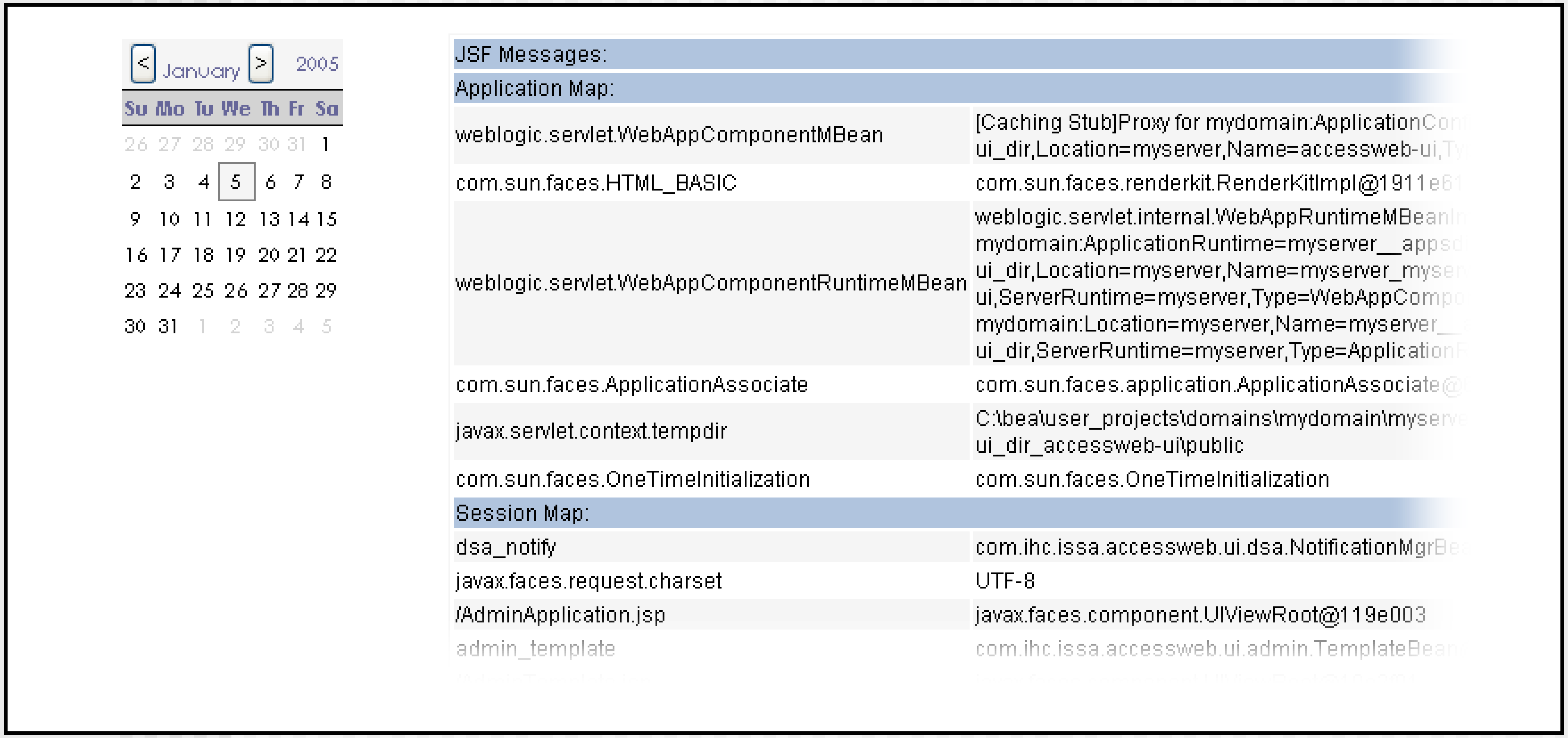
Applicaiton Field Display

Select Checkboxes

Name: [✓ One] ☐ Two ☐ Three

UI Components (Custom)

Some custom JavaServer Faces Components



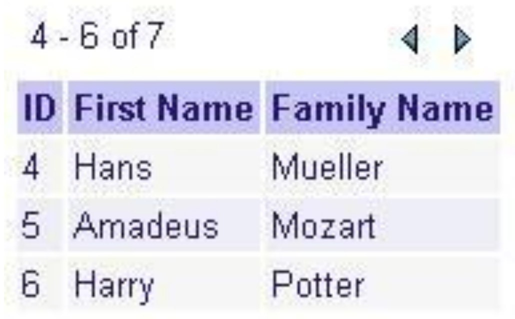
< January > 2005

Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

JSF Messages:	
Application Map:	
weblogic.servlet.WebAppComponentMBean	[Caching Stub]Proxy for mydomain:ApplicationCon ui_dir,Location=myserver,Name=accessweb-ui,Ty
com.sun.faces.HTML_BASIC	com.sun.faces.renderkit.RenderKitImpl@1911e61
weblogic.servlet.WebAppComponentRuntimeMBean	weblogic.servlet.internal.WebAppRuntimeMBeanI mydomain:ApplicationRuntime=myserver__appsd ui_dir,Location=myserver,Name=myserver_myser ui,ServerRuntime=myserver,Type=WebAppCompo mydomain:Location=myserver,Name=myserver__ ui_dir,ServerRuntime=myserver,Type=ApplicationR
com.sun.faces.ApplicationAssociate	com.sun.faces.application.ApplicationAssociate@
javax.servlet.context.tempdir	C:\bea\user_projects\domains\mydomain\myserv ui_dir_accessweb-ui\public
com.sun.faces.OneTimeInitialization	com.sun.faces.OneTimeInitialization
Session Map:	
dsa_notify	com.ihc.issa.accessweb.ui.dsa.NotificationMgrBe
javax.faces.request.charset	UTF-8
/AdminApplication.jsp	javax.faces.component.UIViewRoot@119e003
admin_template	com.ihc.issa.accessweb.ui.admin.TemplateBean

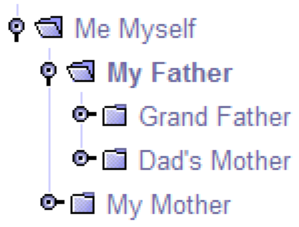
UI Components (Open Source)

Some open source JavaServer Faces Components

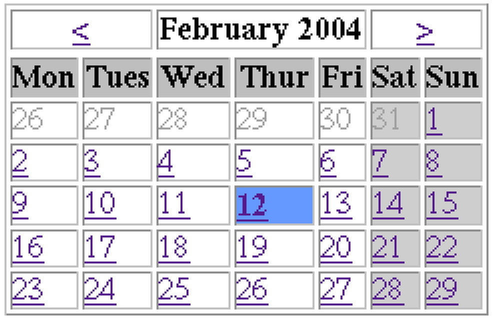


4 - 6 of 7

ID	First Name	Family Name
4	Hans	Mueller
5	Amadeus	Mozart
6	Harry	Potter



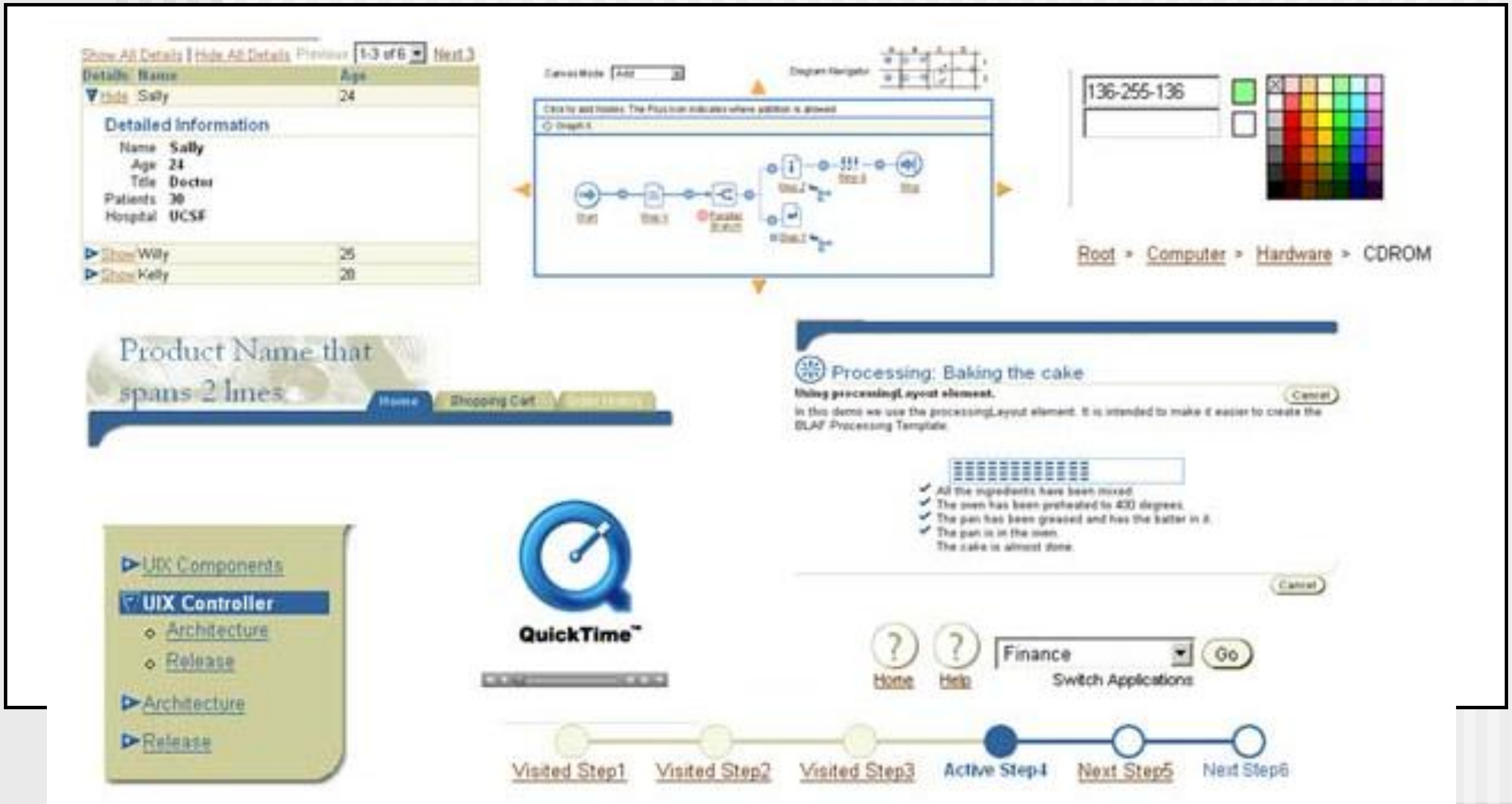
- Me Myself
  - **My Father**
    - Grand Father
    - Dad's Mother
  - My Mother



<	February 2004					>
Mon	Tues	Wed	Thur	Fri	Sat	Sun
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	**12**	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

UI Components (Third Party)

Some third-party JavaServer Faces Components



Internationalization and accessibility

- Components and Converters support multiple locales
- Resource Bundle Support
  - `<f:loadBundle basename="resources" var="res"/>`
  - Define supported locales

```
<application>
  <message-bundle>**resources**</message-bundle>
  <locale-config>
      <default-locale>**en**</default-locale>
      <supported-locale>**fr**</supported-locale>
      <supported-locale>**es**</supported-locale>
  </locale-config>
```

Custom tag library

- You have already seen it
  - html-basic.tld **`<h:.../>`**
  - jsf-core.tld **`<f:.../>`**

Deployment

- BEA WebLogic Server 8.1 SP3
- The JavaServer Faces required jars
  - commons-beanutils.jar
  - commons-collections.jar
  - commons-digester.jar
  - commons-logging.jar
  - jsf-api.jar
  - jsf-impl.jar
  - jstl.jar (must be version 1.0)
  - standard.jar (must be version 1.0)

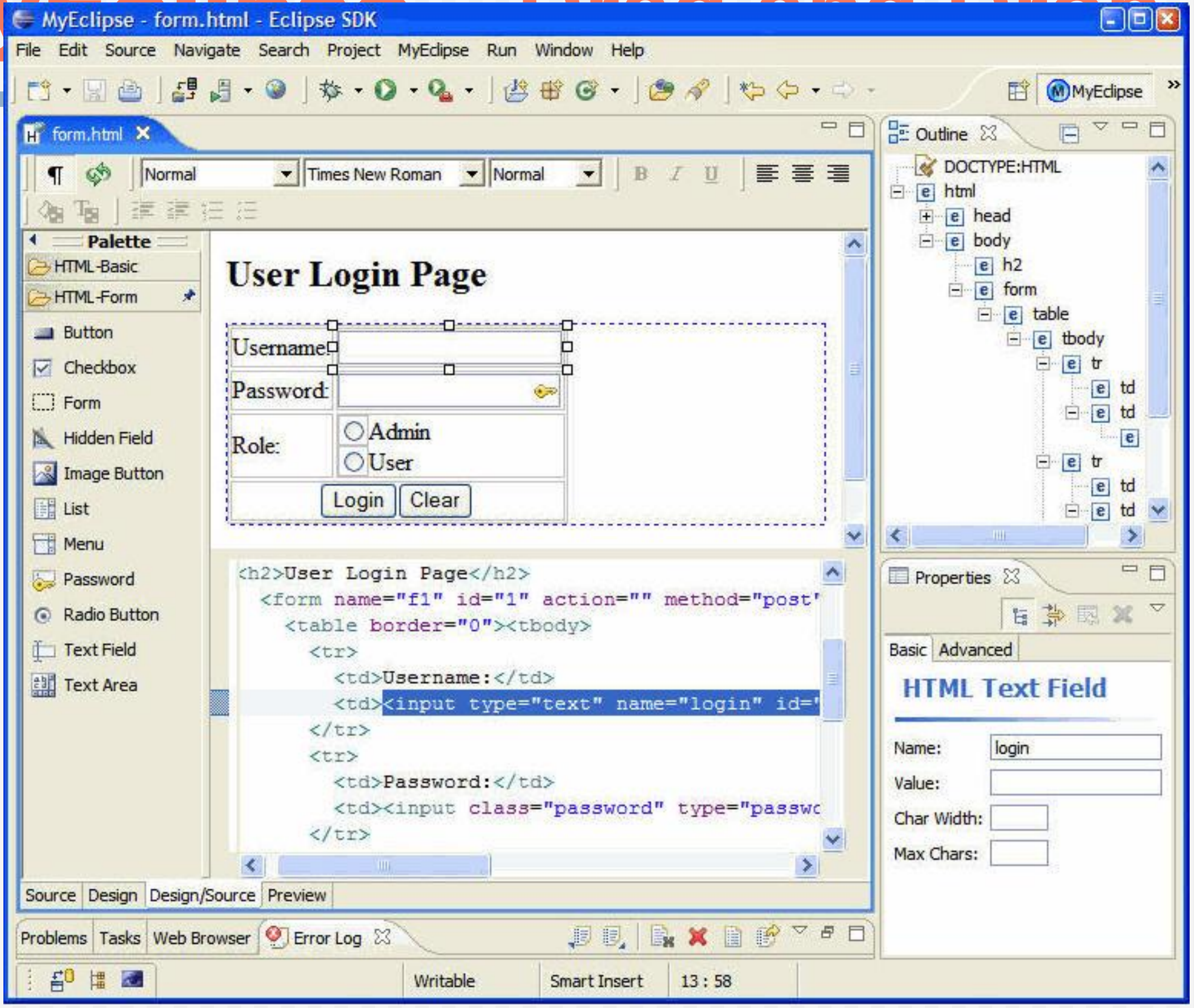
Tool Support

- Most IDEs have limited JSF support

Eclipse*	IntelliJ*	JBuilder	Netbeans*
IBM WSAD	Studio Creator	JDeveloper	Notepad**

- The following IDEs provide the ability to drag and drop components onto a page:
  - IBM WSAD
  - Borland JBuilder
  - Oracle JDeveloper
  - Sun Java Studio Creator

MyEclipse – Drag and Drop



```
<h2>User Login Page</h2>
  <form name="f1" id="1" action="" method="post'
    <table border="0"><tbody>
      <tr>
        <td>Username:</td>
        <td><input type="text" name="login" id="
      </tr>
      <tr>
        <td>Password:</td>
        <td><input class="password" type="passwc
      </tr>
```

Name:	login
Value:	
Char Width:	
Max Chars:	

Political Support

- Sun
- Oracle
- IBM
- …and a bunch of little people but after those three it really doesn’t matter

Component-based Framework

- Introduction
- Java Web Aplication Framework
  - Request-based Framework
  - Component-based Framework
    - Jave Server Faces (JSF)
    - **Tapestry**
    - **Wicket**
  - Hybrid – Meta Framework
  - RIA-based Framework
- Conclusion

Component-based Framework

- Introduction
- Java Web Aplication Framework
  - Request-based Framework
  - Component-based Framework
  - **Hybrid – Meta Framework**
    - RIFE
    - Spring Framework
  - RIA-based Framework
- Conclusion

Component-based Framework

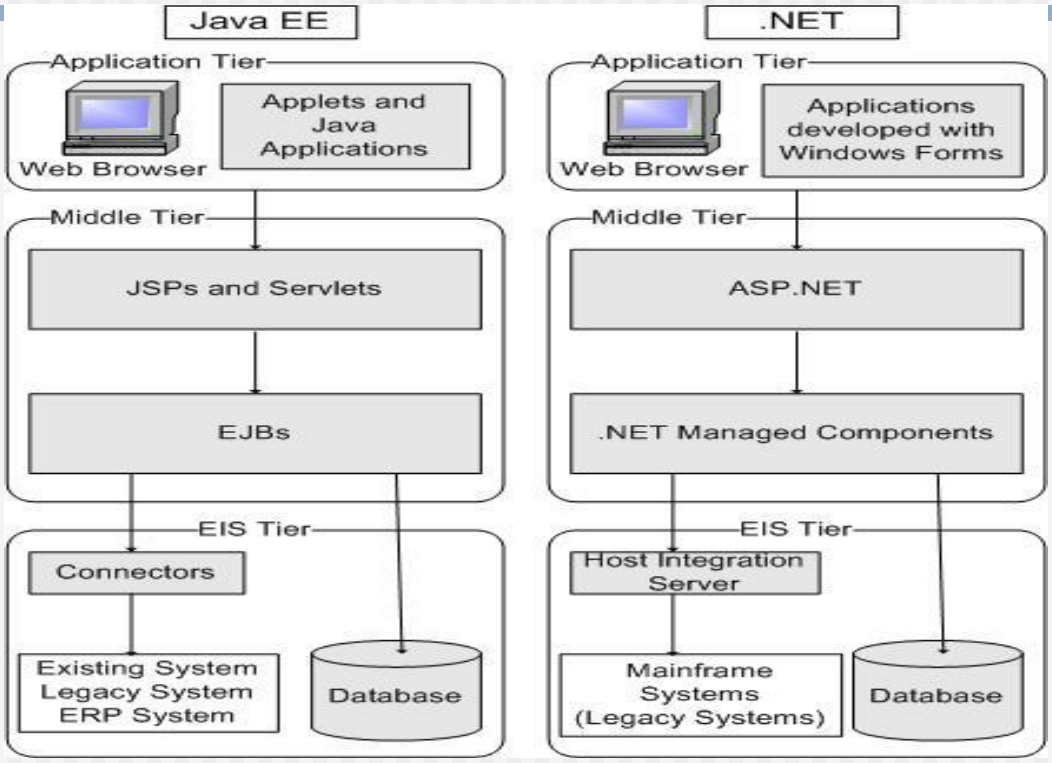
- Introduction
- Java Web Aplication Framework
  - Request-based Framework
  - Component-based Framework
  - Hybrid – Meta Framework
  - **RIA-based Framework**

- Conclusion

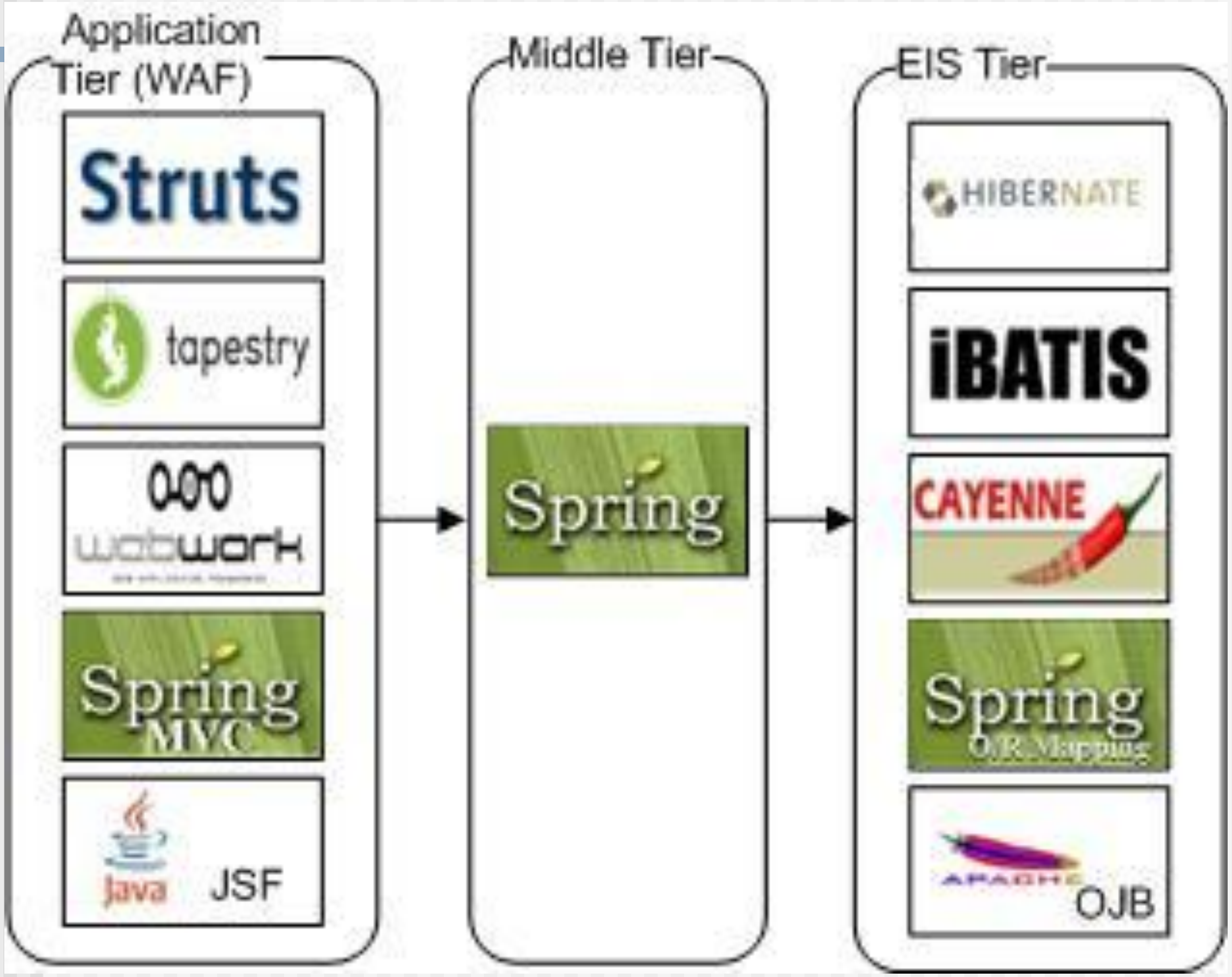
Component-based Framework

- Introduction
- Java Web Aplication Framework
  - Request-based Framework
  - Component-based Framework
  - Hybrid – Meta Framework
  - **RIA-based Framework**
    - DWR
    - Echo2
    - JSON-RPC-Java
- Conclusion

Java EE and .NET tiers



Java EE frameworks


Application Tier (WAF)
Struts
tapestry
webwork
Spring MVC
Java
JSF
Middle Tier
Spring
EIS Tier
HIBERNATE
iBATIS
CAYENNE
Spring O/R Mapping
APACHE
OJB