WEB SERVICE FRAMEWORKS

Pushpendra Singh

Disclaimer: these slides have been prepared by using contents from resources mentioned in the reference slide

Developing Web Services

- Servlets
- JAX-RS 2.0
- Spring MVC
- …

Disclaimer: these slides have been prepared by using contents from resources mentioned in the reference slide

Servlets

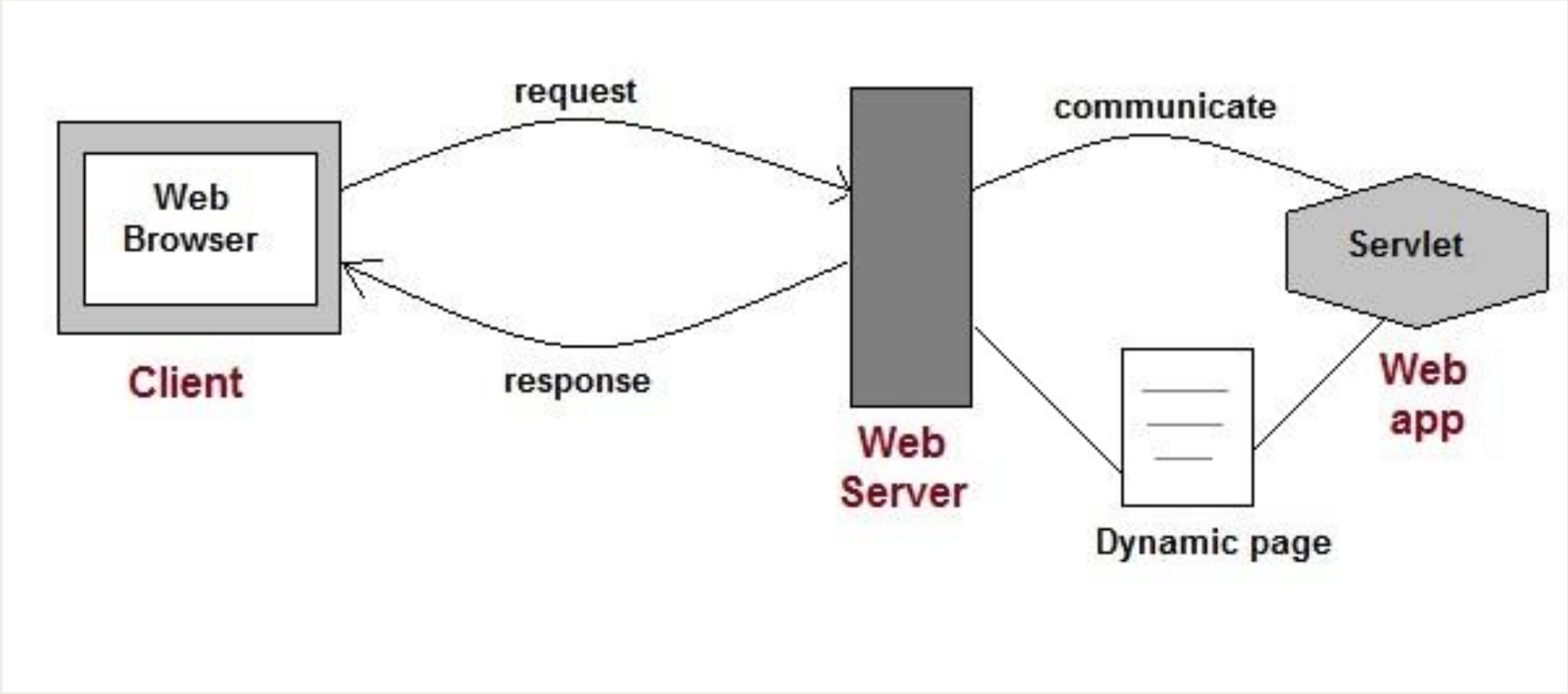


image source: http://www.studytonight.com/servlet/introduction-to-servlet.php

Before Servlets

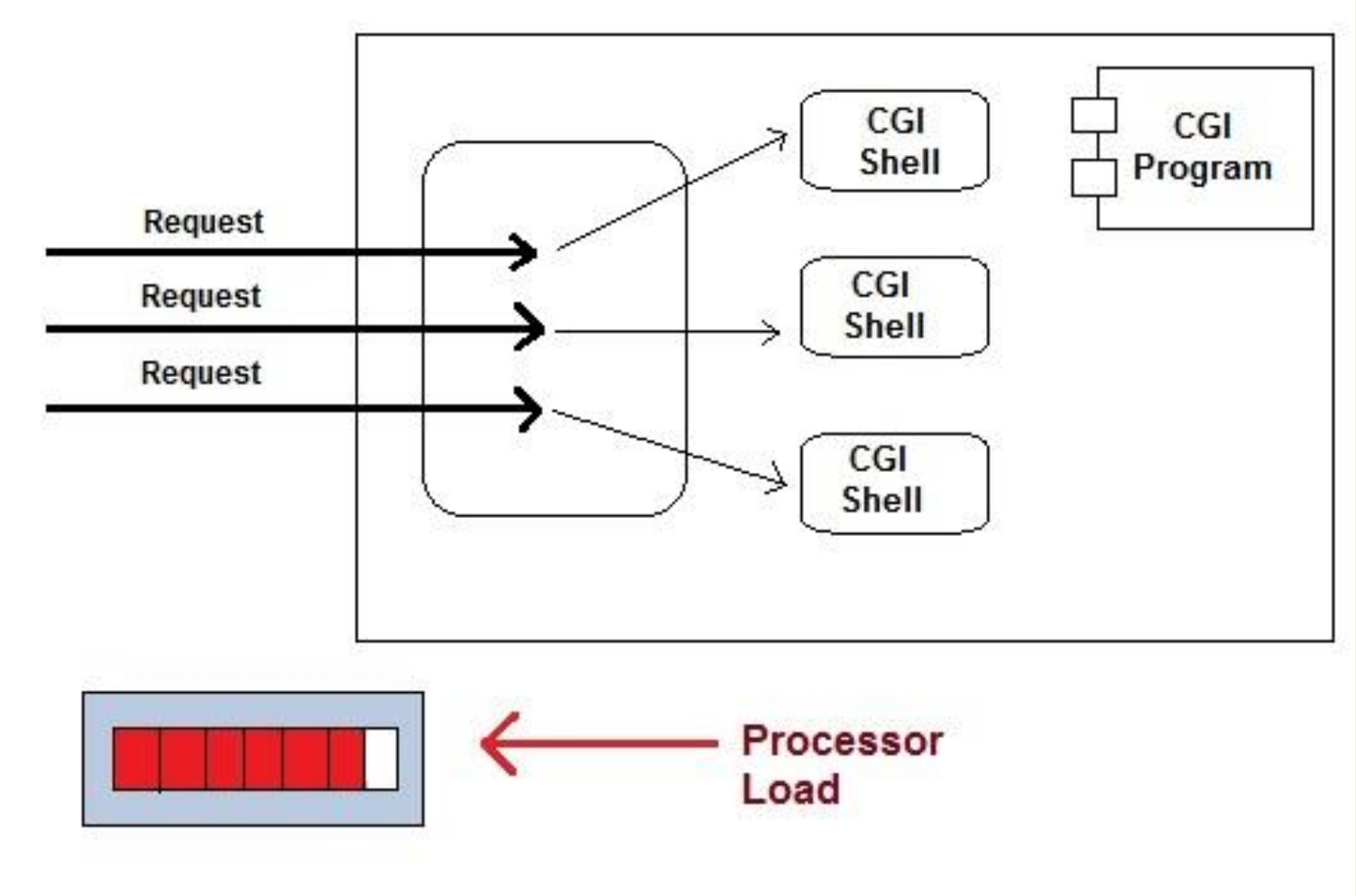


image source: http://www.studytonight.com/servlet/introduction-to-servlet.php

Servlets

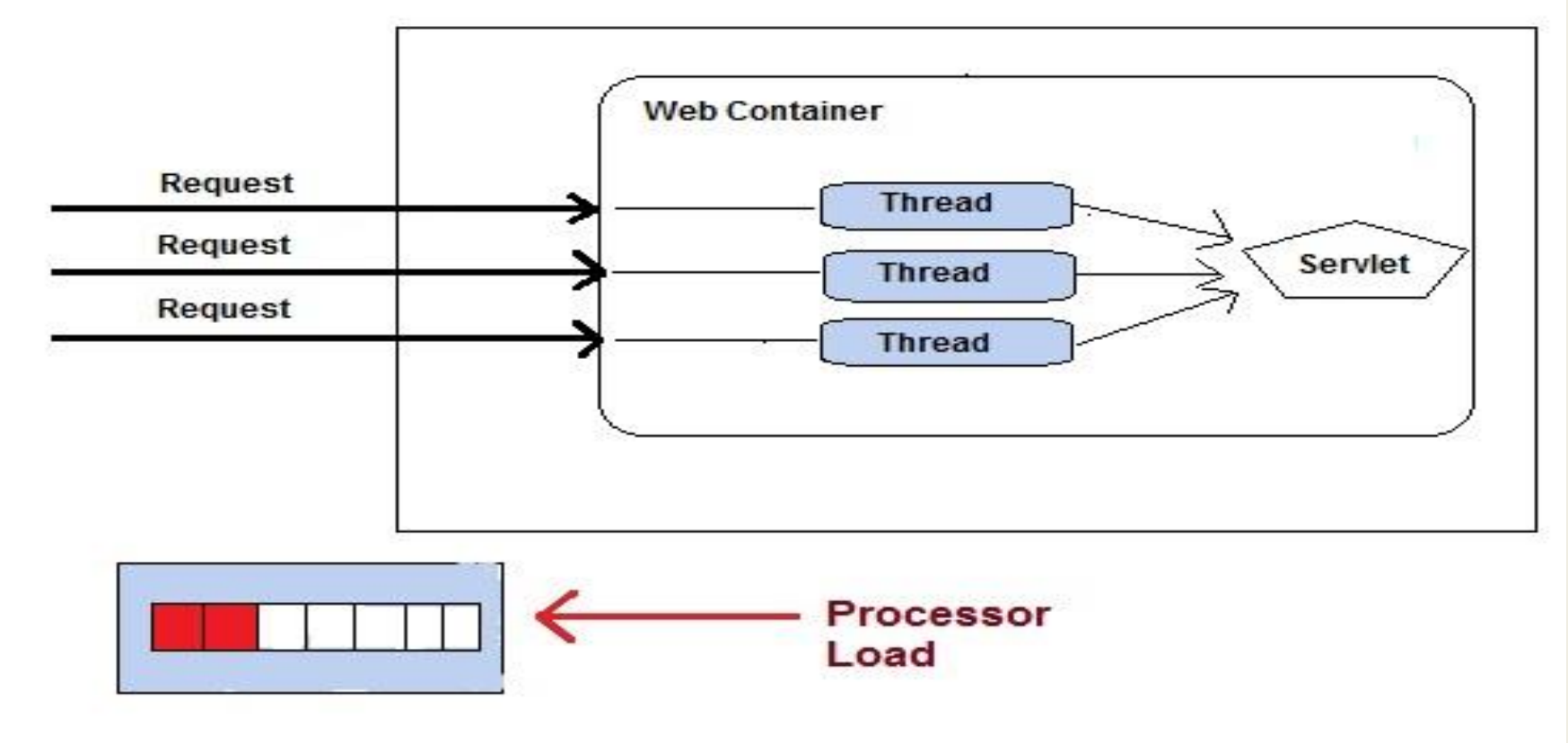


image source: http://www.studytonight.com/servlet/introduction-to-servlet.php

Servlets

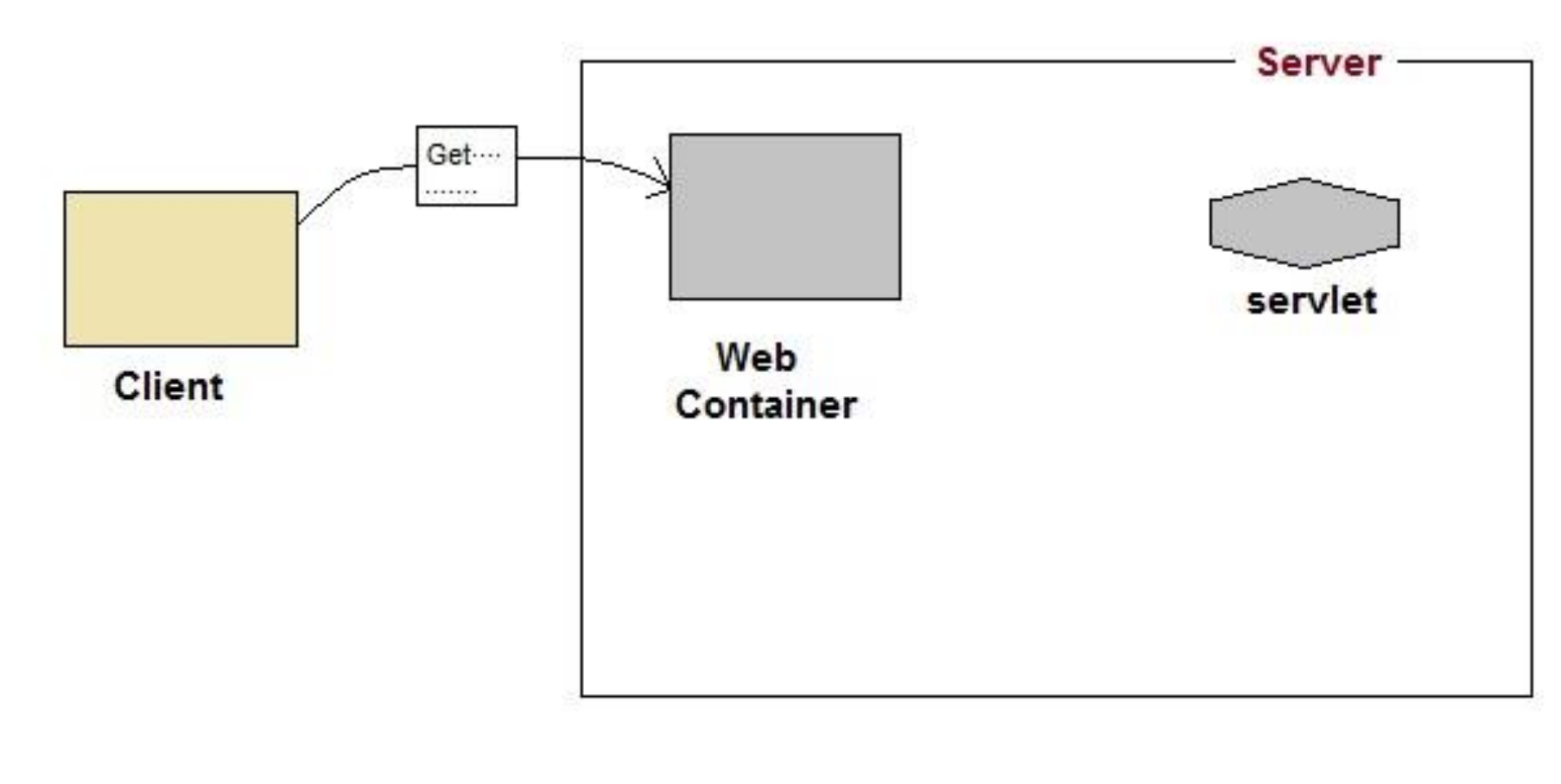


image source: http://www.studytonight.com/servlet/introduction-to-servlet.php

Servlets

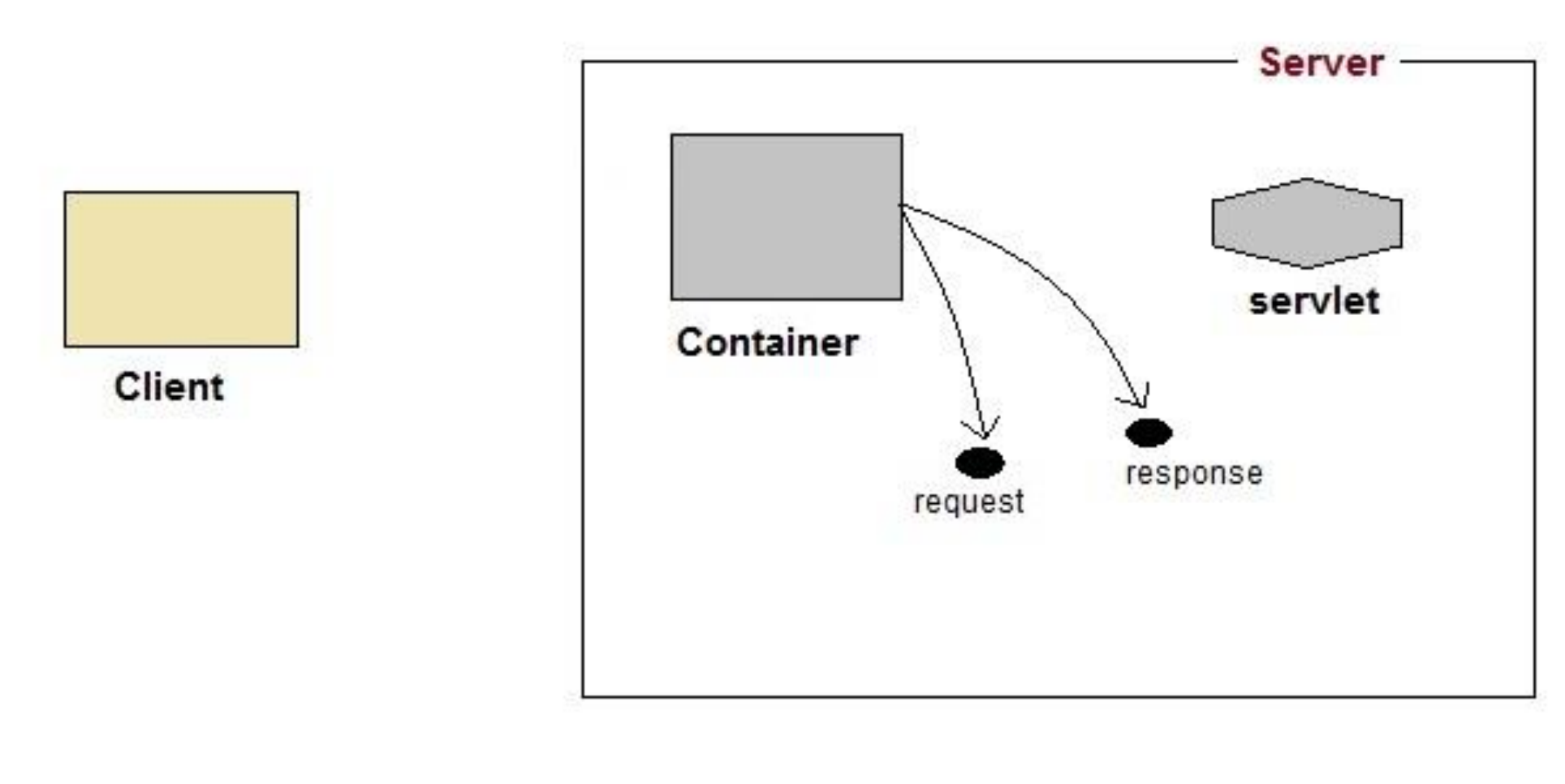


image source: http://www.studytonight.com/servlet/introduction-to-servlet.php

Servlets

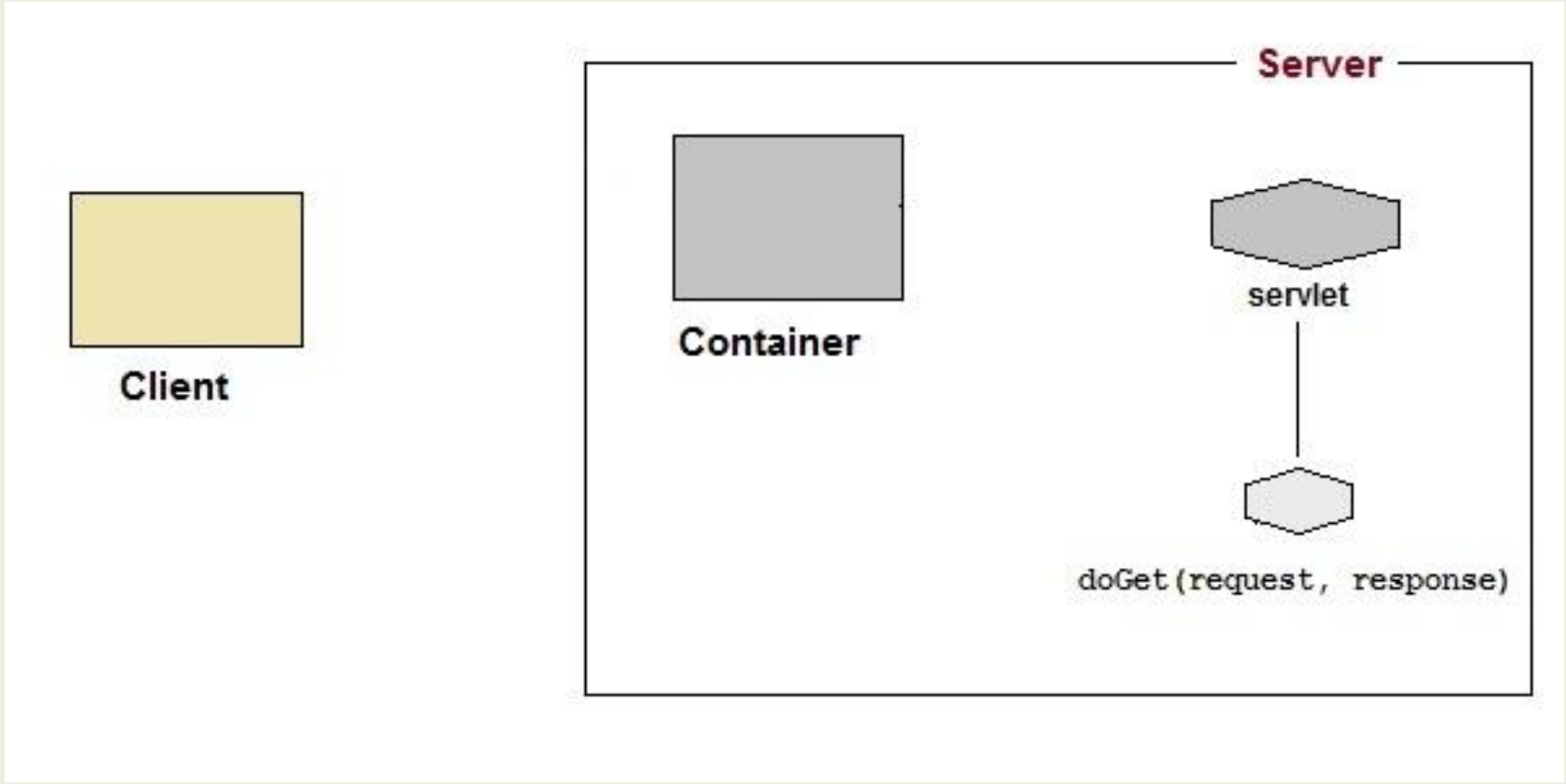


image source: http://www.studytonight.com/servlet/introduction-to-servlet.php

Servlets

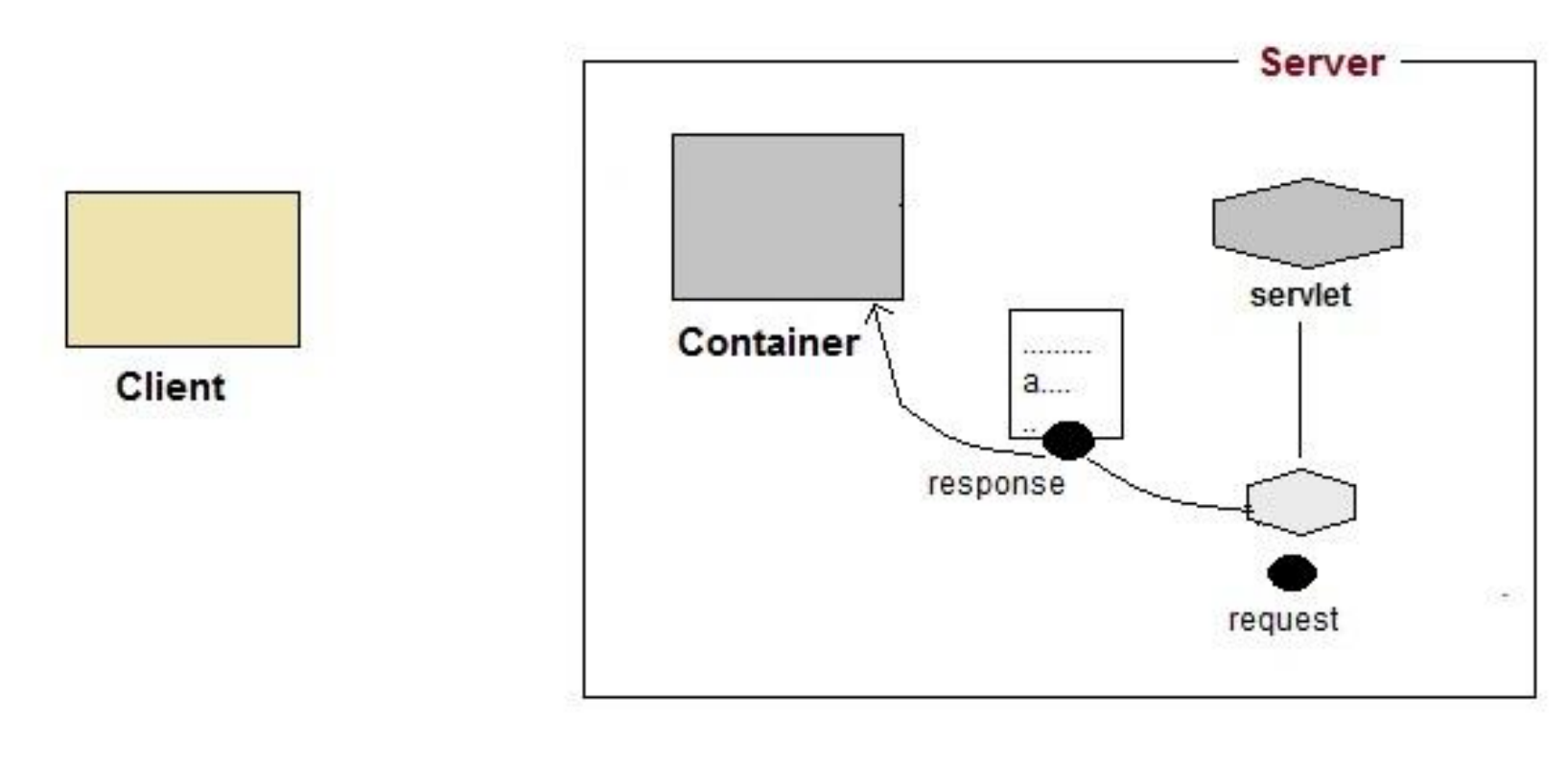


image source: http://www.studytonight.com/servlet/introduction-to-servlet.php

Servlets

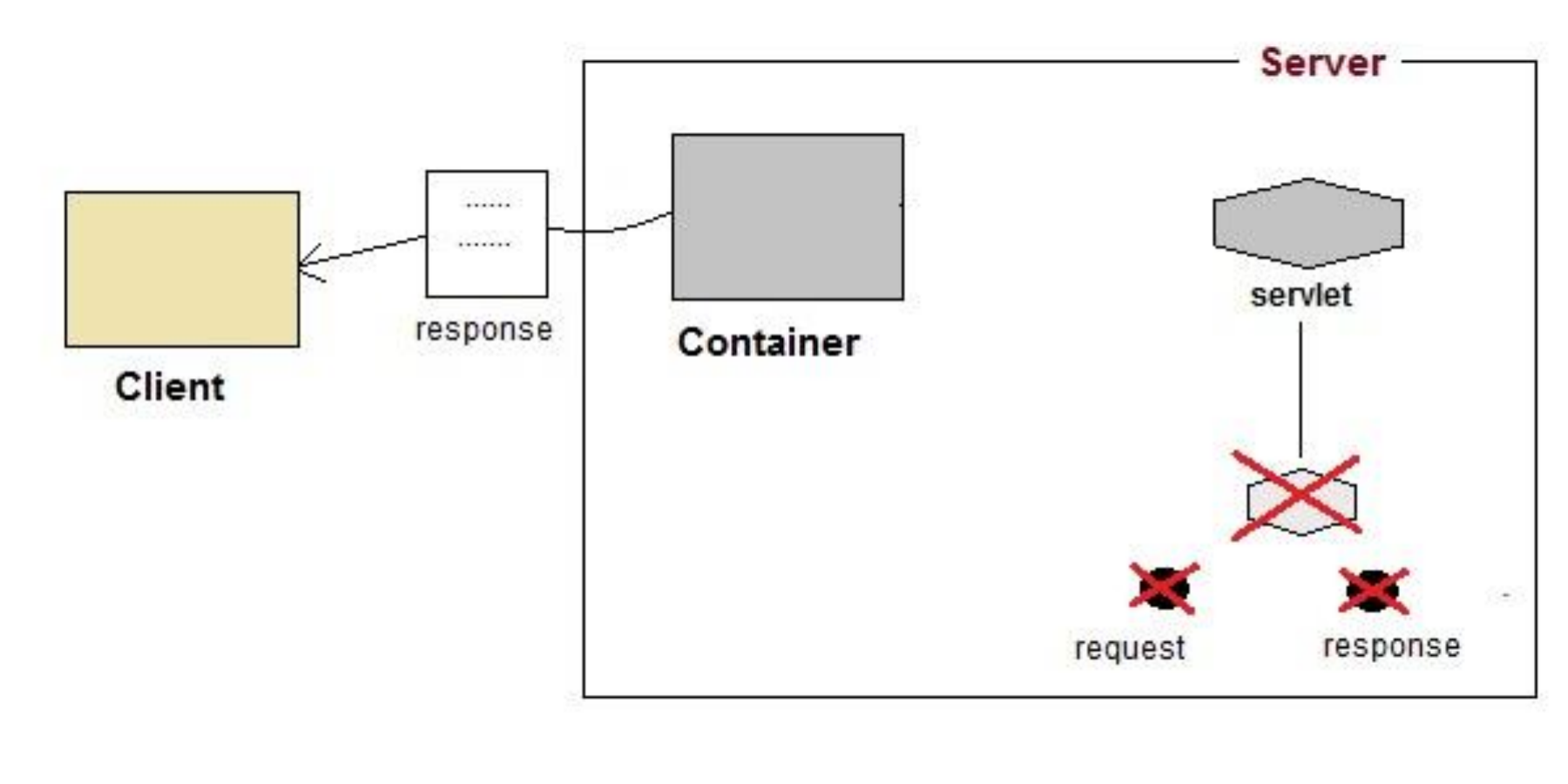


image source: http://www.studytonight.com/servlet/introduction-to-servlet.php

Servlets

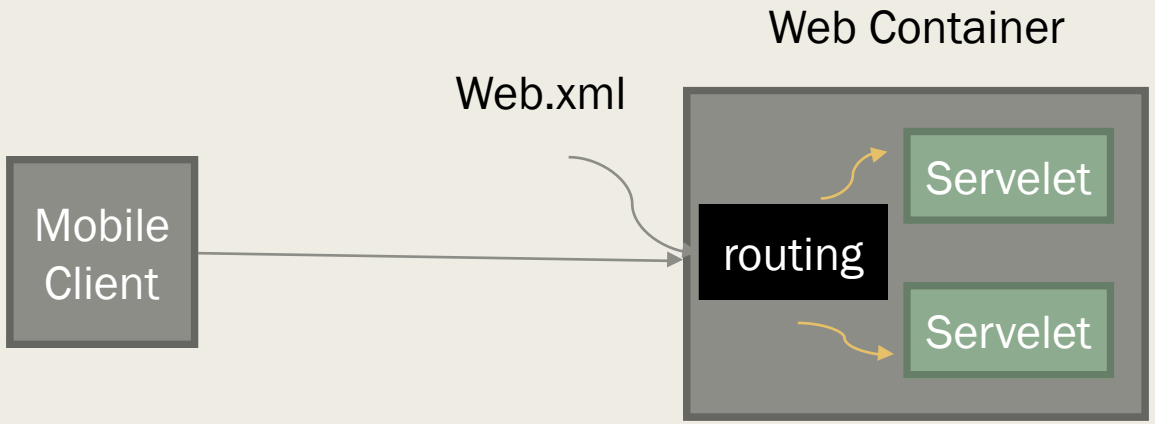


image source: http://www.studytonight.com/servlet/introduction-to-servlet.php

Routing and Web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"  version="2.4"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/dtd/..._2_3.dtd">
  <servlet>
    <servlet-name>video</servlet-name>
    <servlet-class>org.mobilecloud.VideoServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>video</servlet-name>
    <url-pattern>/video</url-pattern>
  </servlet-mapping>

</web-app>
```

image source: http://www.studytonight.com/servlet/introduction-to-servlet.php

Servlet

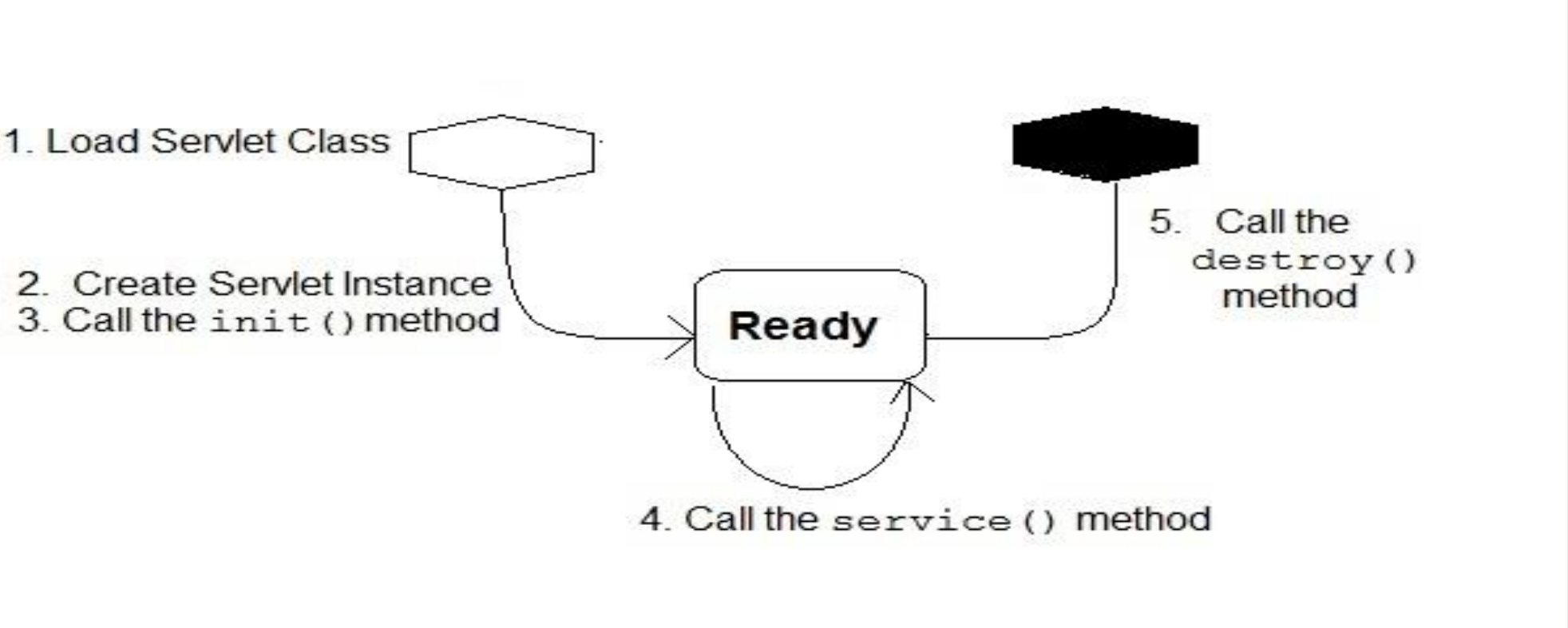


image source: http://www.studytonight.com/servlet/introduction-to-servlet.php

JAX-RS Service

- Defined in 2008 to simplify RESTful service implementation
- Uses Java annotations
- Other features to support HTTP use

Disclaimer: these slides have been prepared by using contents from resources mentioned in the reference slide

Annotations

```
@secured
public void storeVideo(…)
{
 …
}
```

- Annotation: meta-data providing extra information
- Works as a modifier
- Require a processing tool that understands the annotations

Annotations

- Usage
  - *Automatic generation of auxiliary files, e.g. deployment descriptors*
  - *Automatic generation of code for testing, logging, etc.*

Annotations

Each annotation has the format

@AnnotationName(element=value,…)

Example:

```
@secured(row=“user”)
public @interface secured
{
 String row() default “tiger”;
}
```

Standard Annotations

- Annotations for Compilation
  - *Deprecated, SuppressWarnings, Override, Generated*
- Annotations for Managing Resources
  - *PostConstruct, PreDestroy, Resource, Resources*
- Meta-Annotations
  - *Target, Retention (Souce/Class/Runtime), Documented, Inherited*

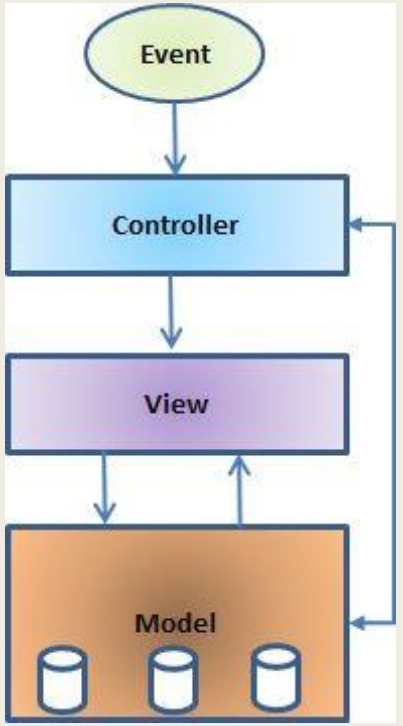
Spring MVC

- A Framework to create web services
- Uses MVC Architecture
  - *Model*
  - *View*
  - *Controller*
- Forces Inversion of Control

Disclaimer: these slides have been prepared by using contents from resources mentioned in the reference slide

MVC Architecture

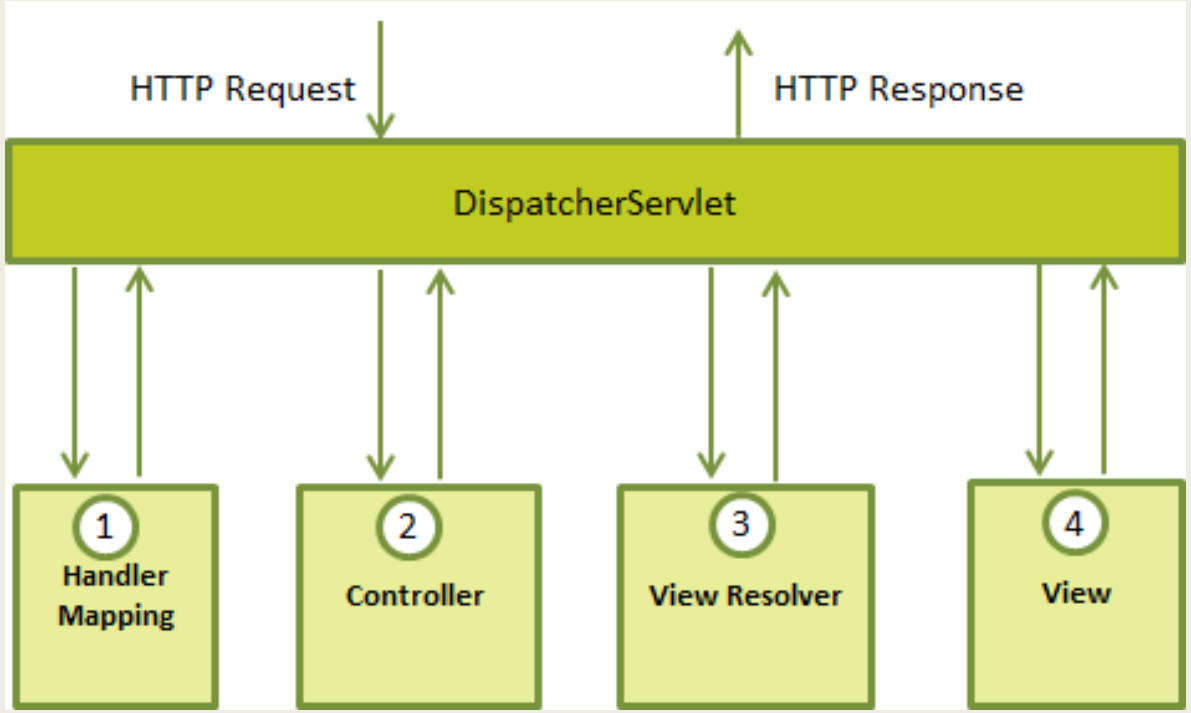
- Model View Controller: design pattern for developing web applications
- Model: responsible for managing data
- View: responsible for displaying data
- Controller: application logic



Source: Head First Design Patterns, image source: http://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm

Spring MVC Framework

- The DispatcherServlet



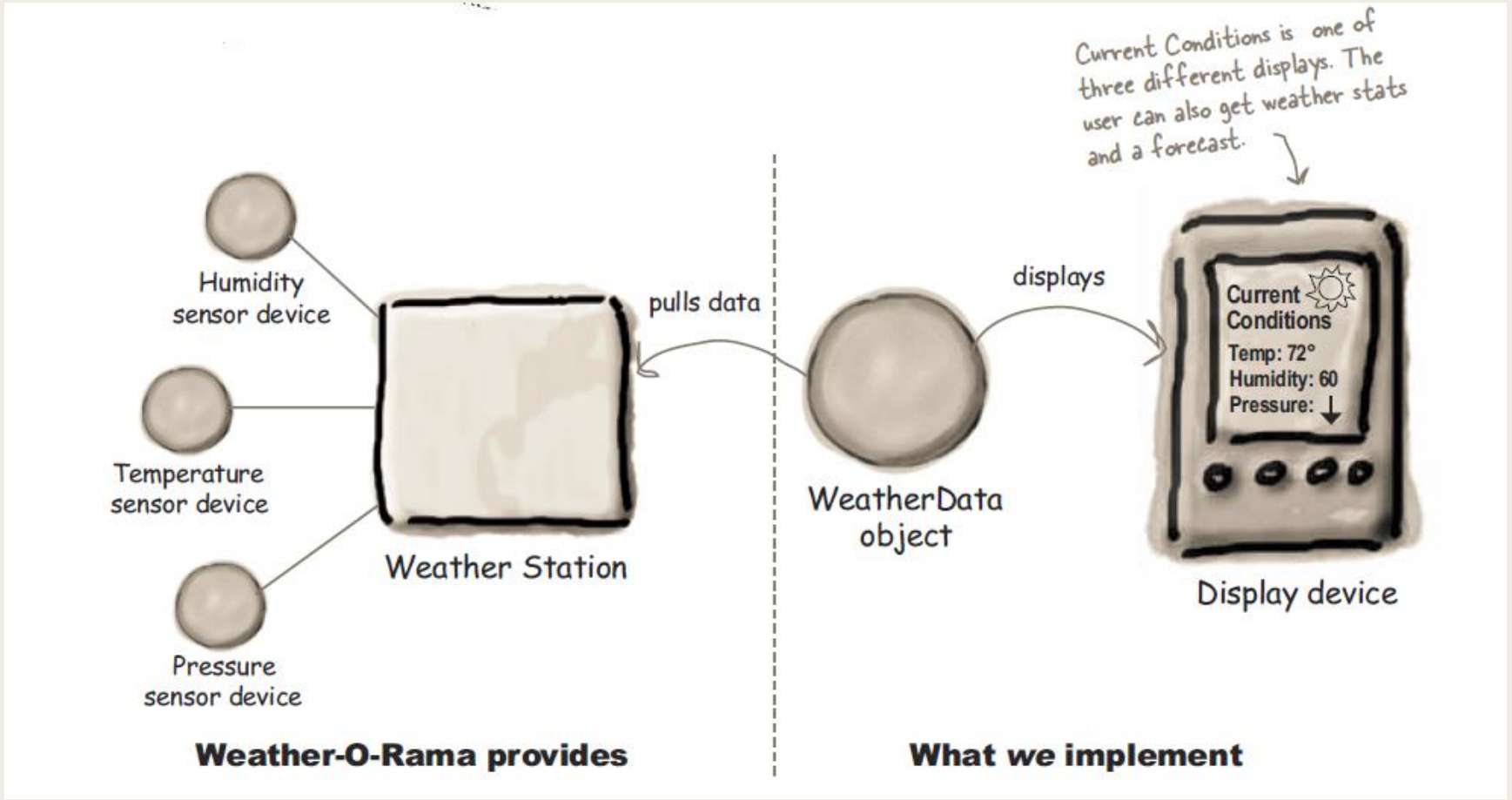
Source: Head First Design Patterns, http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

Observer Pattern

- Do not miss out when something interesting happens

Source: Head First Design Patterns

A Weather Monitoring Application



Source: Head First Design Patterns

A Weather Monitoring Application

- The WeatherData object knows how we talk to the physical Weather Station to get updated data
- The Weather data object then updates its displays for the three different display element:
  - *Current Conditions (temp., humidity, pressure)*
  - *Weather Statistics*
  - *Forecast*

Source: Head First Design Patterns

WeatherData Class

- getTemp()
- getHumidity()
- getPressure()

- measurementsChanged()
  - *this method gets called whenever measurements change*

Source: Head First Design Patterns

Implementation

- Implement measurementsChanged()
- Implement three displays()
  - *currentConditionsDisplay*
  - *statisticsDisplay*
  - *forcastDisplay*
  - *Always updated whenever there is a new measurement*

- The system must be expendable so that one can add a new display if needed

Source: Head First Design Patterns

```
public void measurementsChanged(){

float temp = getTemp();
float humidity = get Humidity();
float pressure = getPressure();

currentConditionsDisplay.update(temp, humidity, pressure);
statisticsDisplay.update(temp, humidity, pressure);
forecastDisplay.update(temp, humidity, pressure);

}
```

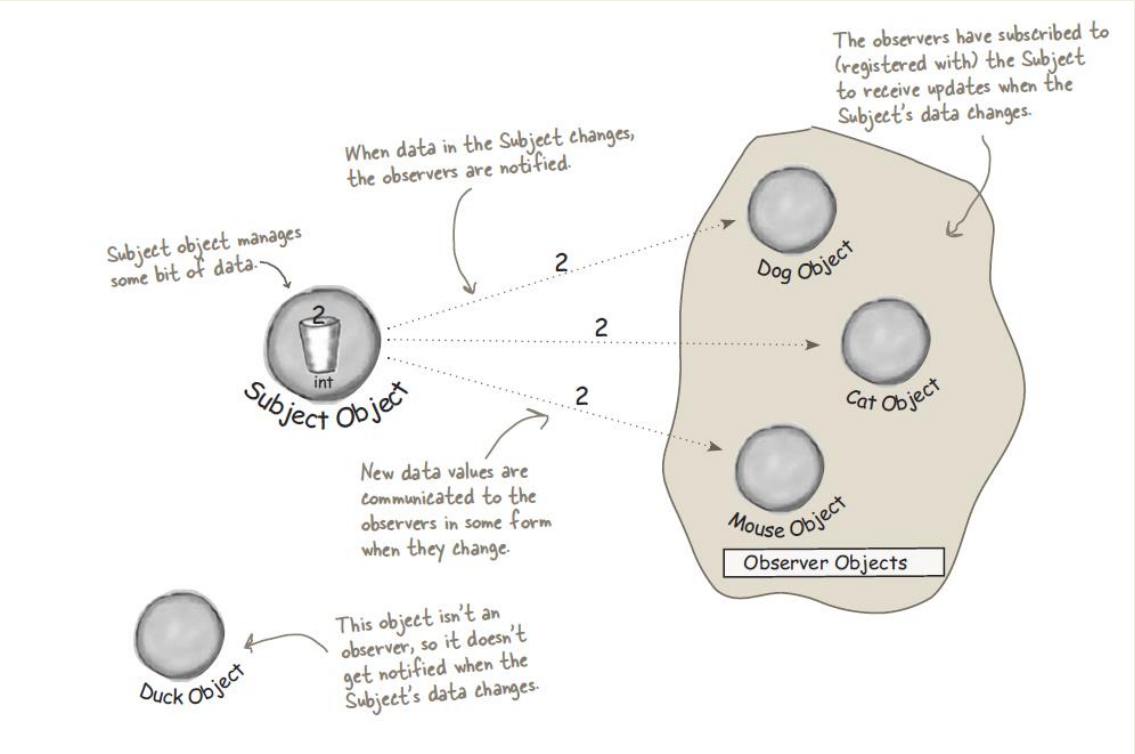
Source: Head First Design Patterns

Observer Pattern

- How newspaper or magazine subscription works
- Newspaper publisher publishes newspapers
- You subscribe to a particular publisher, and every time there's a new edition it get delivered to you. As long as you remain a subscriber, you get new newspapers
- You unsubscribe when you don't want papers anymore and they stop being delivered
- While the publisher remains in business, people and other businesses constantly subscribe/unsubscribe to the newspaper.

Source: Head First Design Patterns

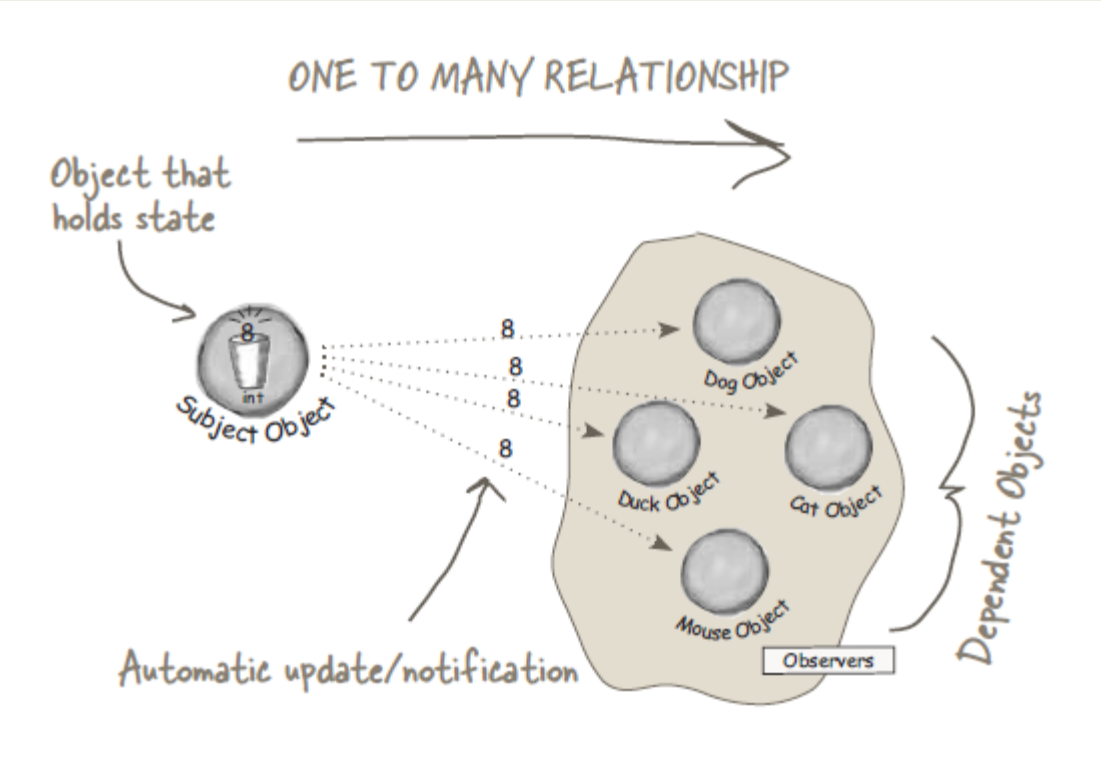
Publishers + subscribers = Observer



Source: Head First Design Patterns

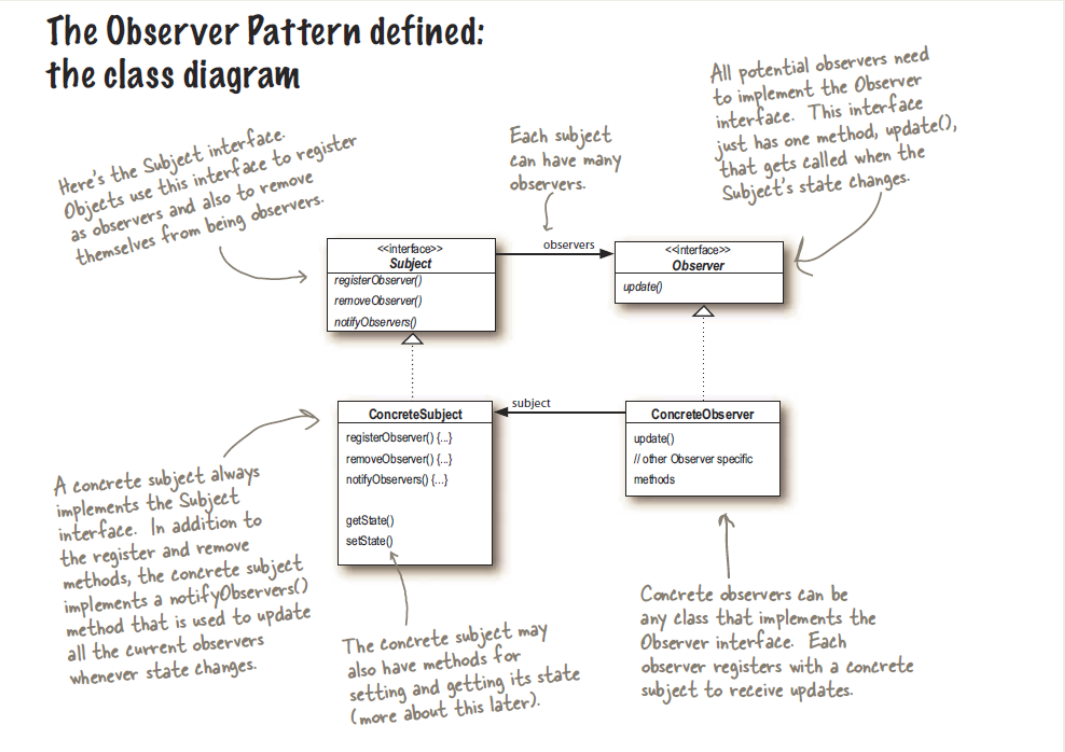
The Observer Pattern

- Defines a one-to-many dependency between objects
- When one object changes state, all of its dependents are notified



Source: Head First Design Patterns

The Observer Pattern



Source: Head First Design Patterns

Loose coupling

- Subjects and observers are loosely coupled
  - *Only thing the subject knows about the observes is that it implements a certain interface*
- We can add/remove observers any time
  - *No need to modify the subject*
- Reuse of subject and observers
  - *Change to either will not affect the other*
    - As long as they implement the interfaces

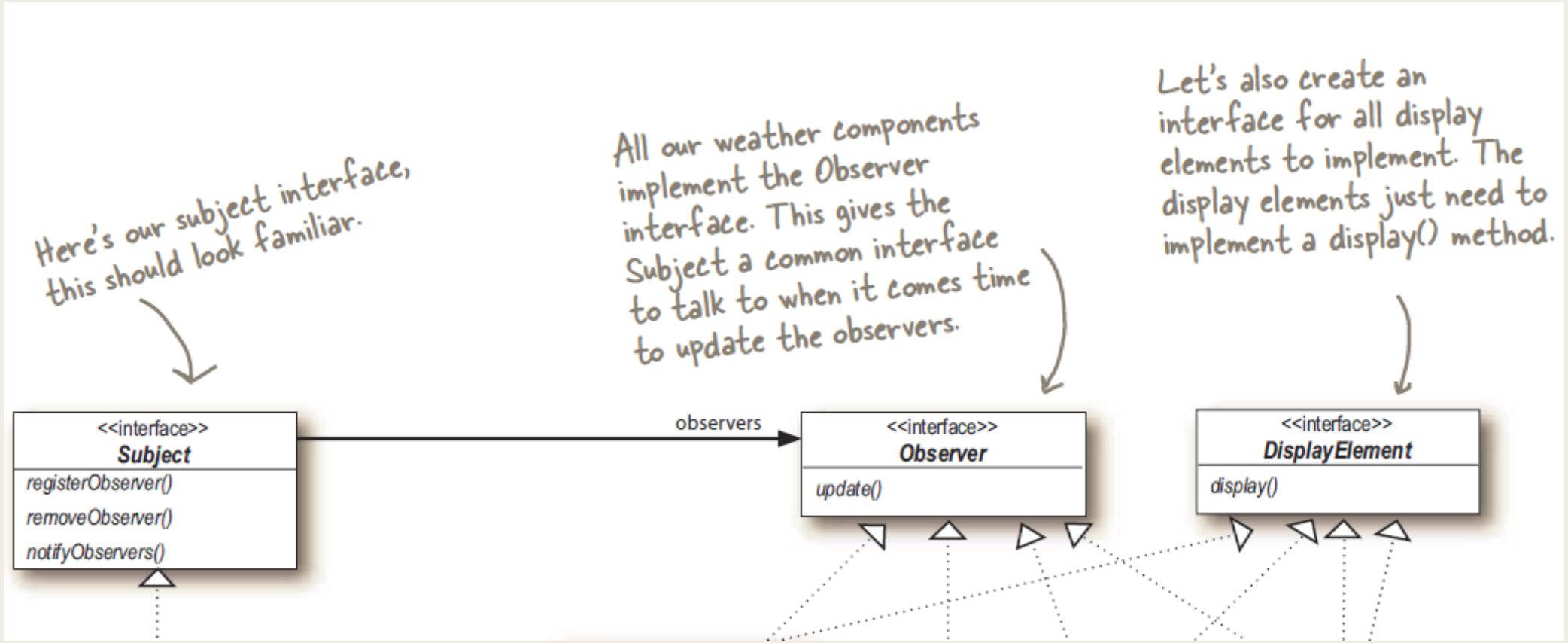
Source: Head First Design Patterns

Weather Application

- WeatherData
  - *One*
  - *Has states – temp., humidity, pressure – that change*
    - whenever there is a change, we need to notify the display elements
- Displays
  - *Many – could be of different types*

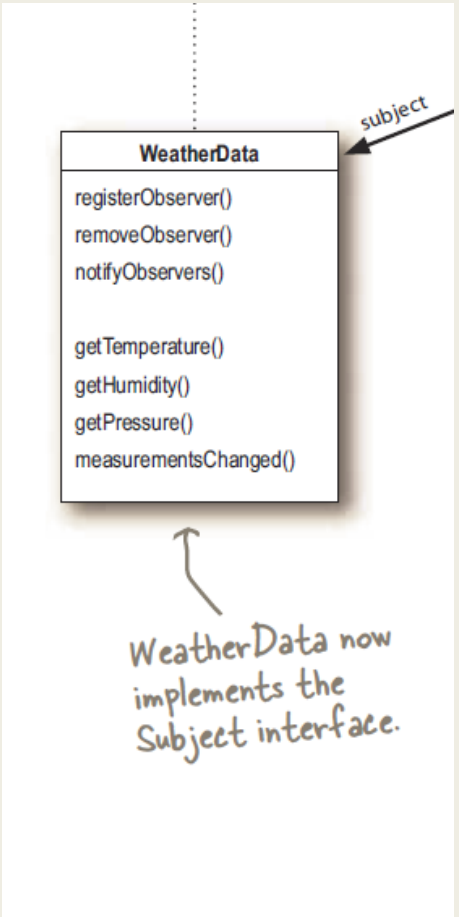
Source: Head First Design Patterns

Designing the Weather Station



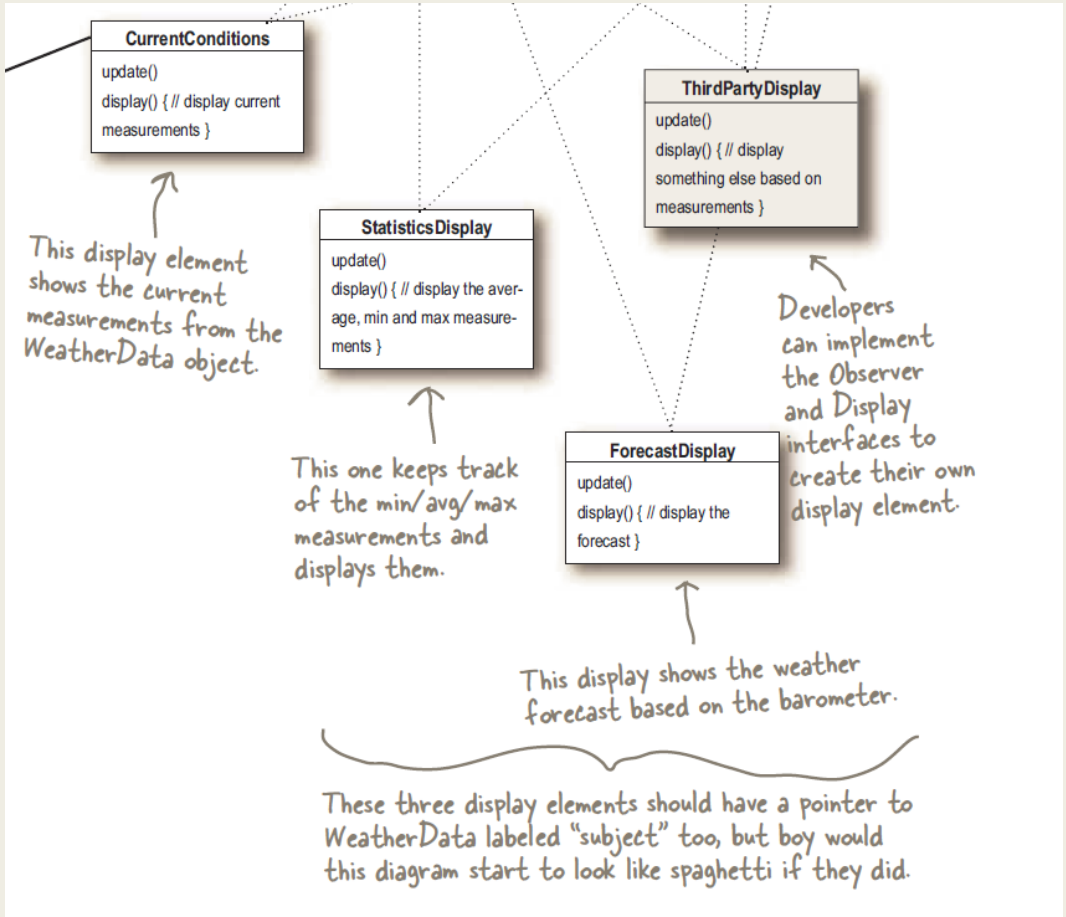
Source: Head First Design Patterns

Designing the Weather Station



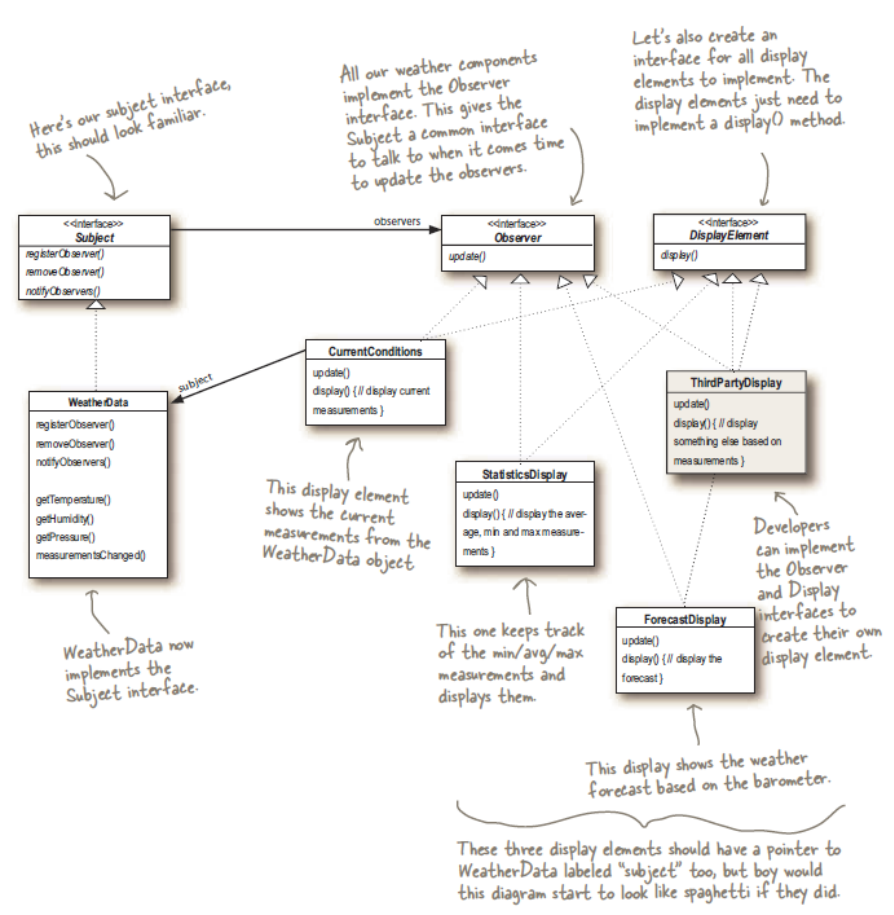
Source: Head First Design Patterns

Designing the Weather Station



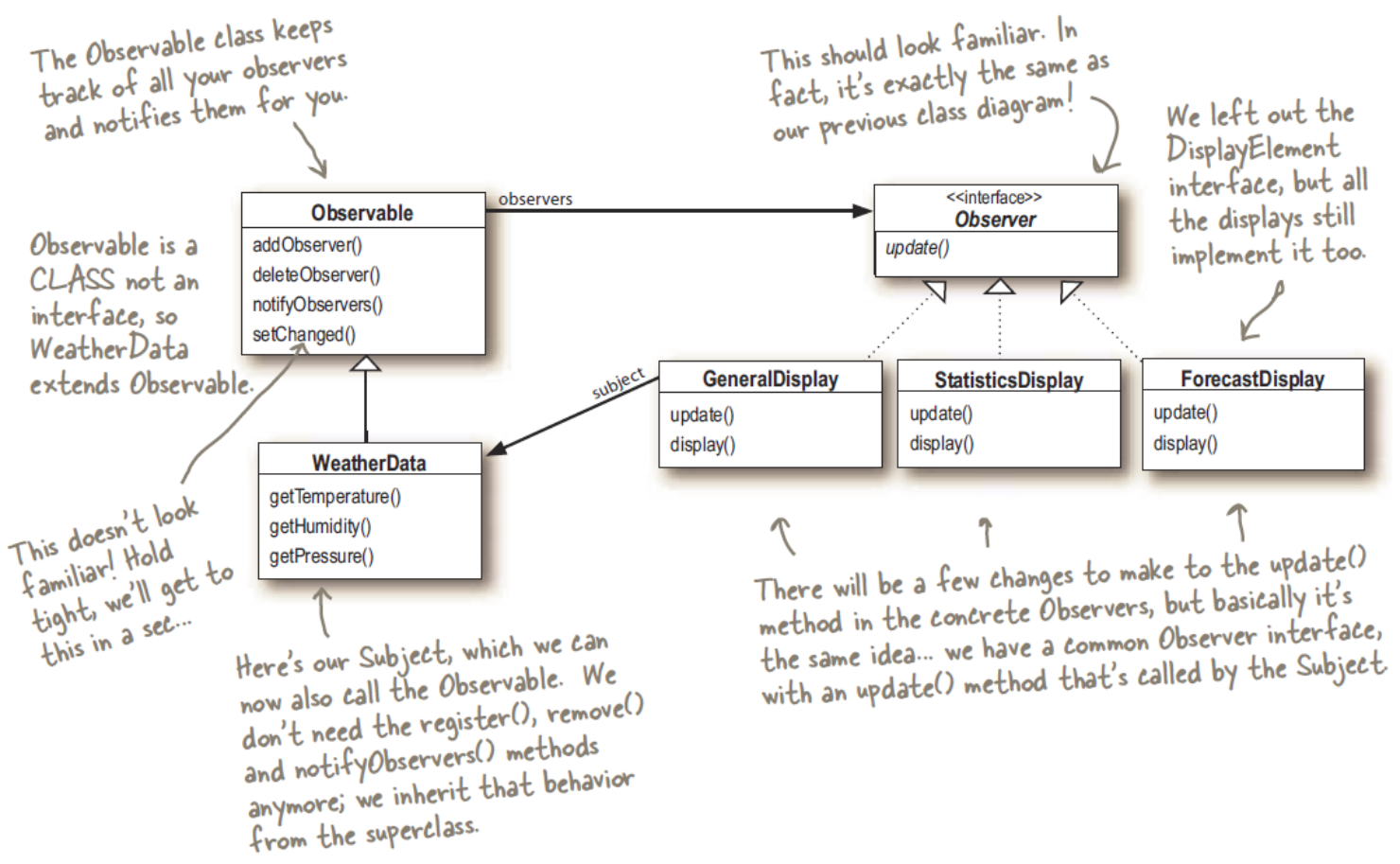
Source: Head First Design Patterns

Designing the Weather Station



Source: Head First Design Patterns

Java's Built-in Observer Pattern



Source: Head First Design Patterns

References

- Burke, Bill (2013-11-12). RESTful Java with JAX-RS 2.0. O'Reilly Media
- Rest in Practice: Hypermedia and Systems Architecture
  - *Publisher: Shroff/O'Reilly*
- RESTful Java with JAX-RS 2.0
  - *Publisher: O'Reilly*
- Developing RESTful Services with JAX-RS 2.0, WebSockets, and JSON
  - *Publisher: PACKT*
- Building a RESTful Web Service with Spring
  - *Publisher: PACKT*
- WebSocket
  - *Publisher: O'Reilly*

Disclaimer: these slides have been prepared by using contents from resources mentioned in the reference slide