Implementing Web Applications

Antonio Goncalves

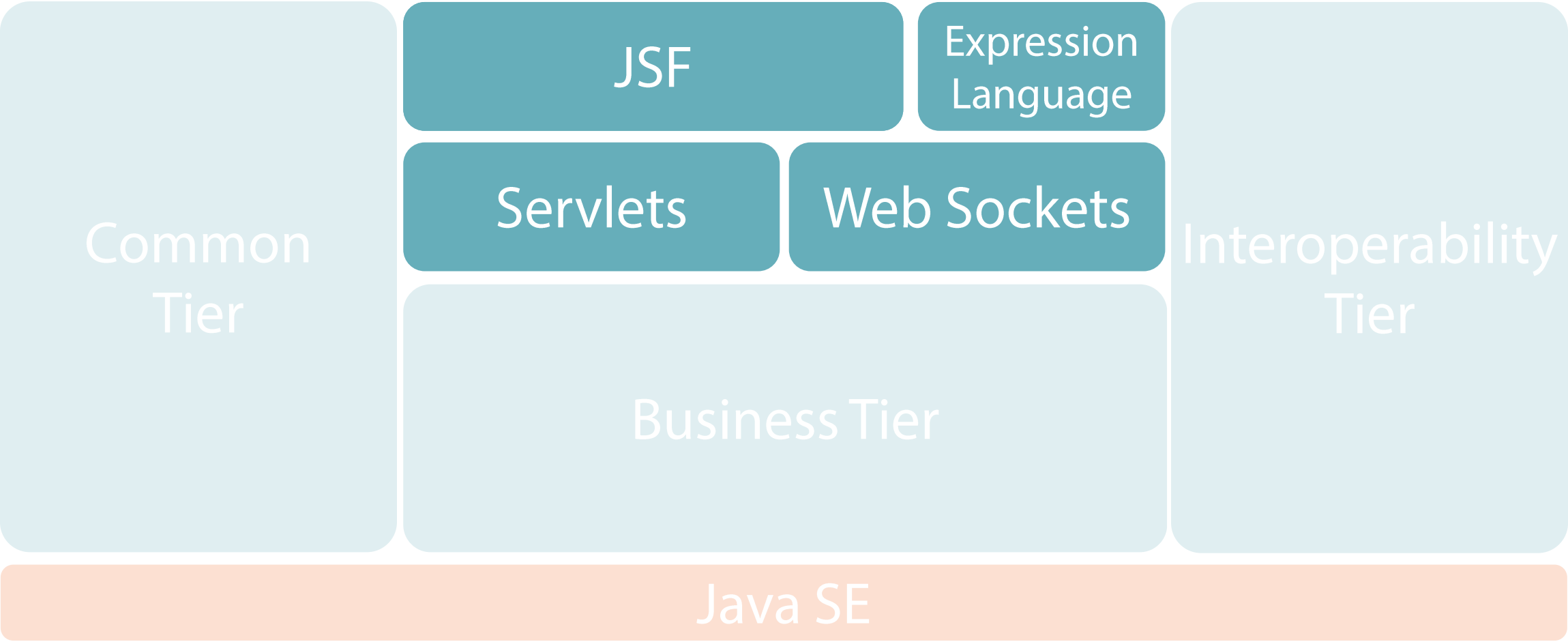
@agoncal | www.antoniogoncalves.org

Previous Module

- Business tier
- Persistence
- Transaction management
- Batch processing

Module Outline

Module Outline



Servlets

Servlet 3.1

What Are Servlets?

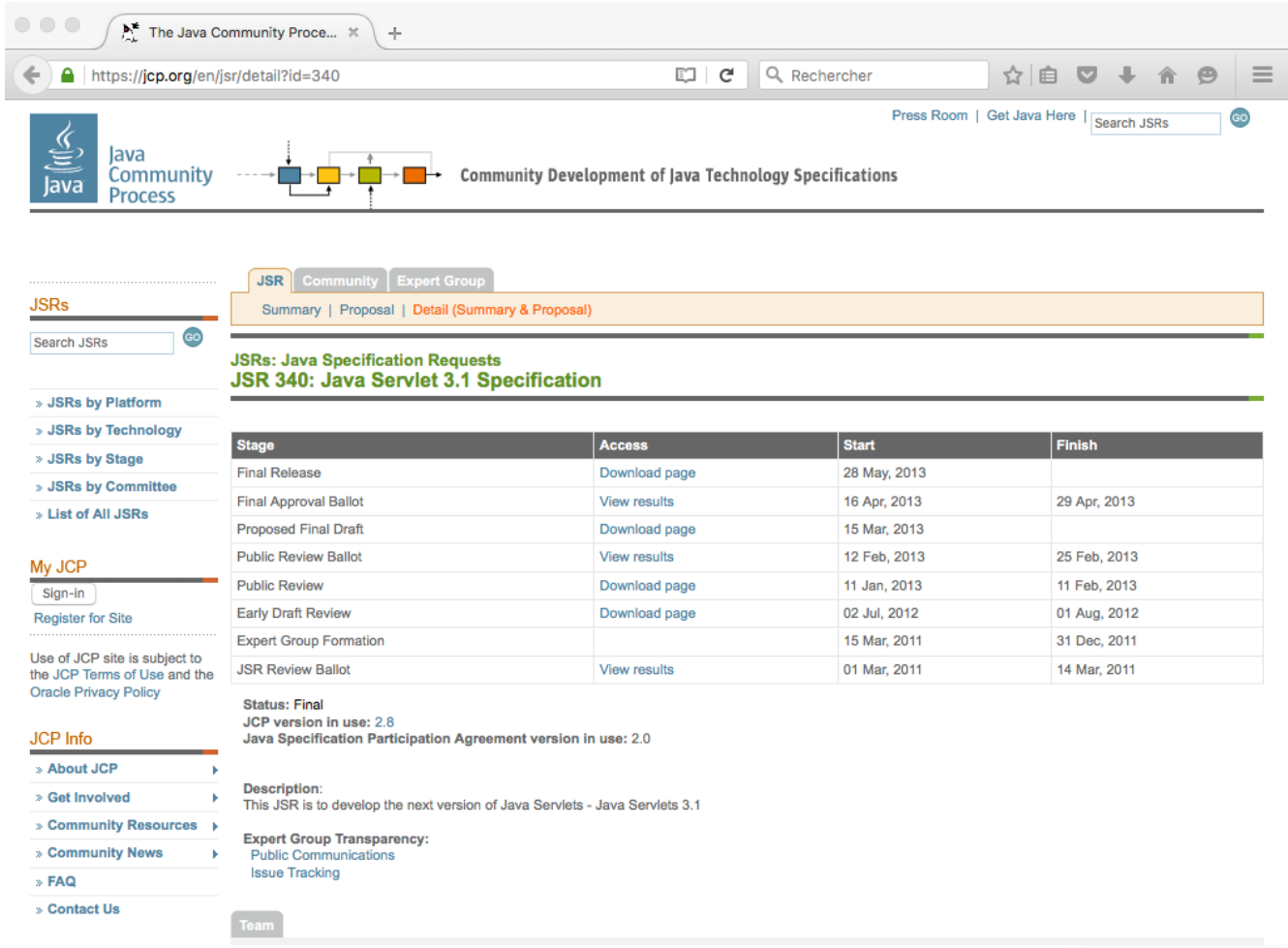
- Web components
- Run on Web container
- Between HTTP client and business tier
- Interact via request/response paradigm
- Platform independent

When to Use Servlets

- Handle HTTP requests
- Collect input from Web users
  - Through Web page forms
  - URL parameters
- Dispatch work to business layer
- Create dynamic Web pages

Servlet Specification

- Servlet 3.1
- JSR 340
- http://jcp.org/en/jsr/detail?id=340



Servlet Implementations

- GlassFish
- Tomcat
- Jetty
- Undertow
- Weblogic
- Websphere

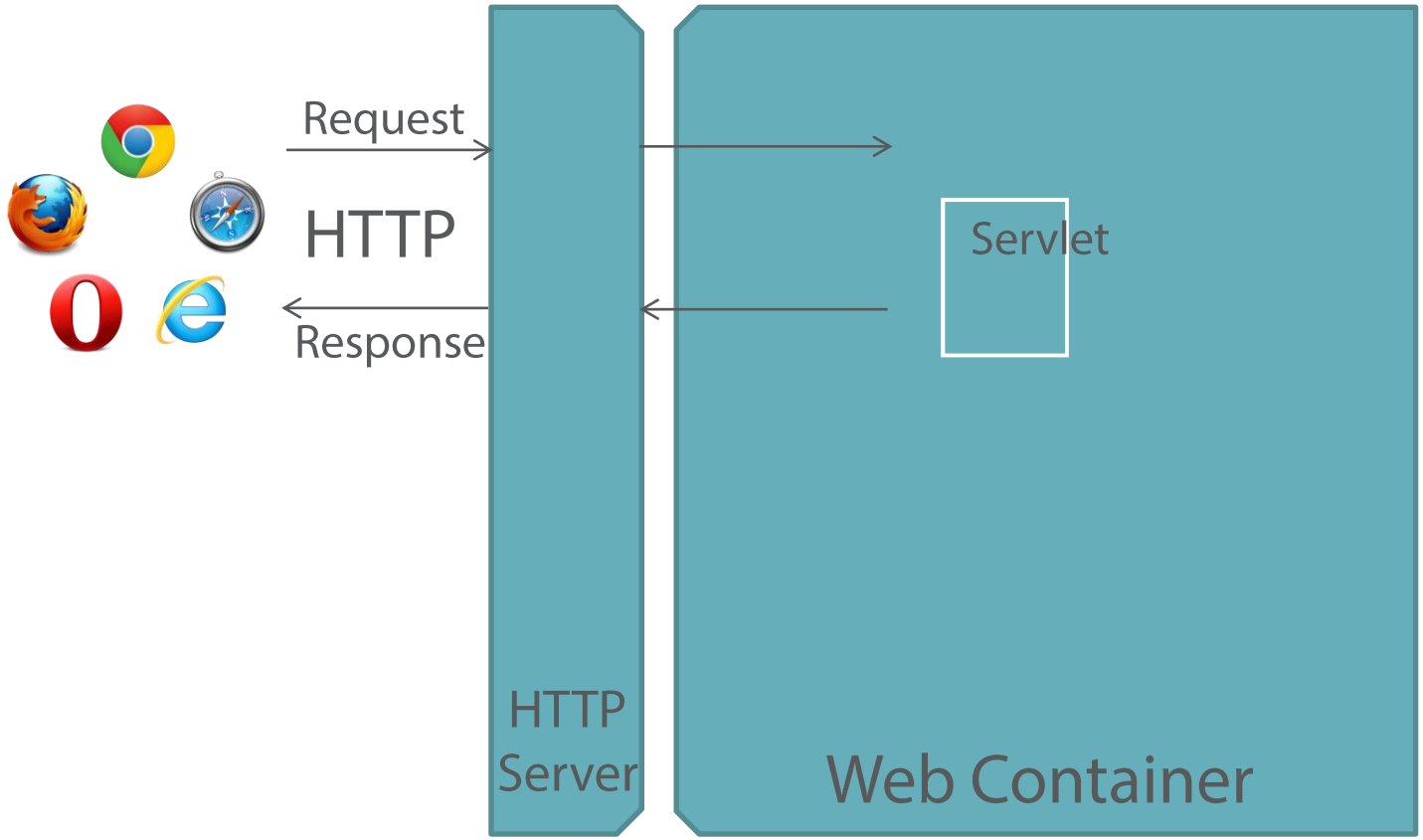
jetty://

undertow

ORACLE®
WEBLOGIC

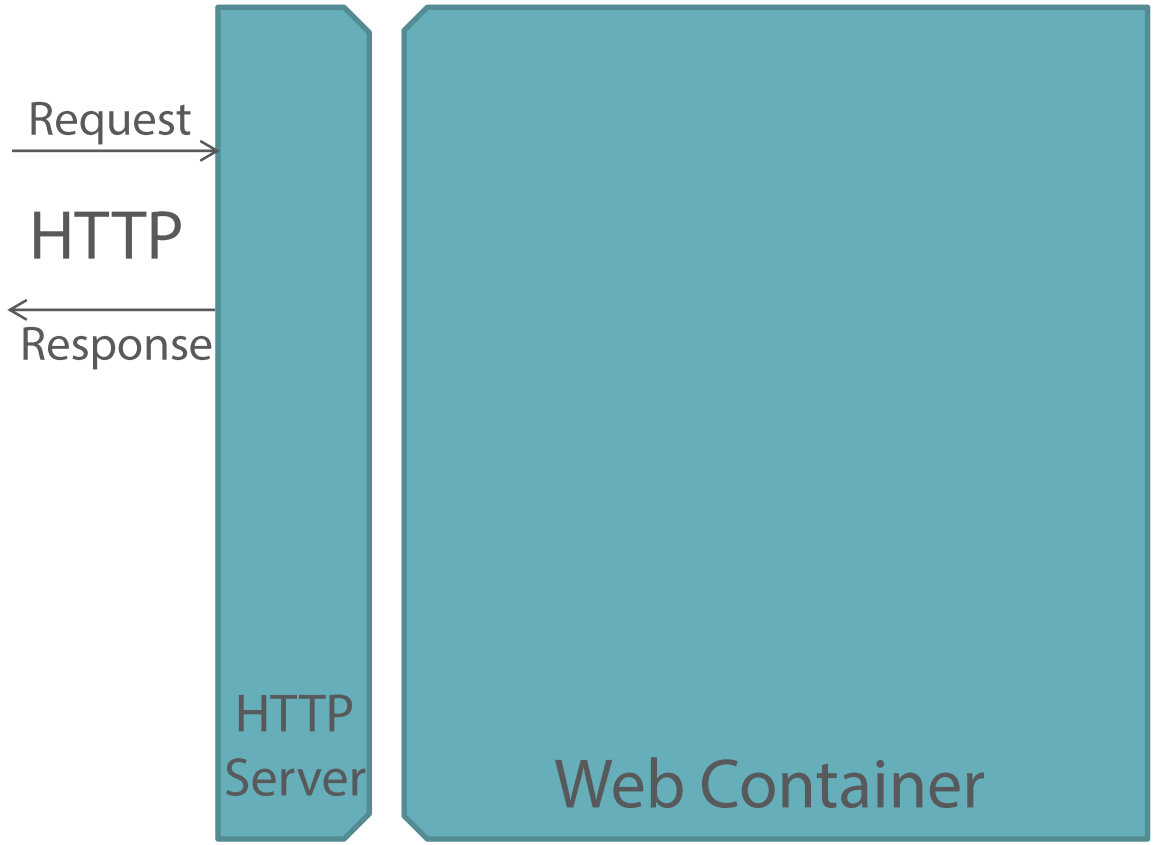
Understanding Servlets

- Web browser
- HTTP Server
- HTTP Protocol
- Web Container
- Servlet



HTTP Server and Web Container

- Run on same or different host
- Network services
- HTTP request
- Loading and instantiating servlets
- HTTP response

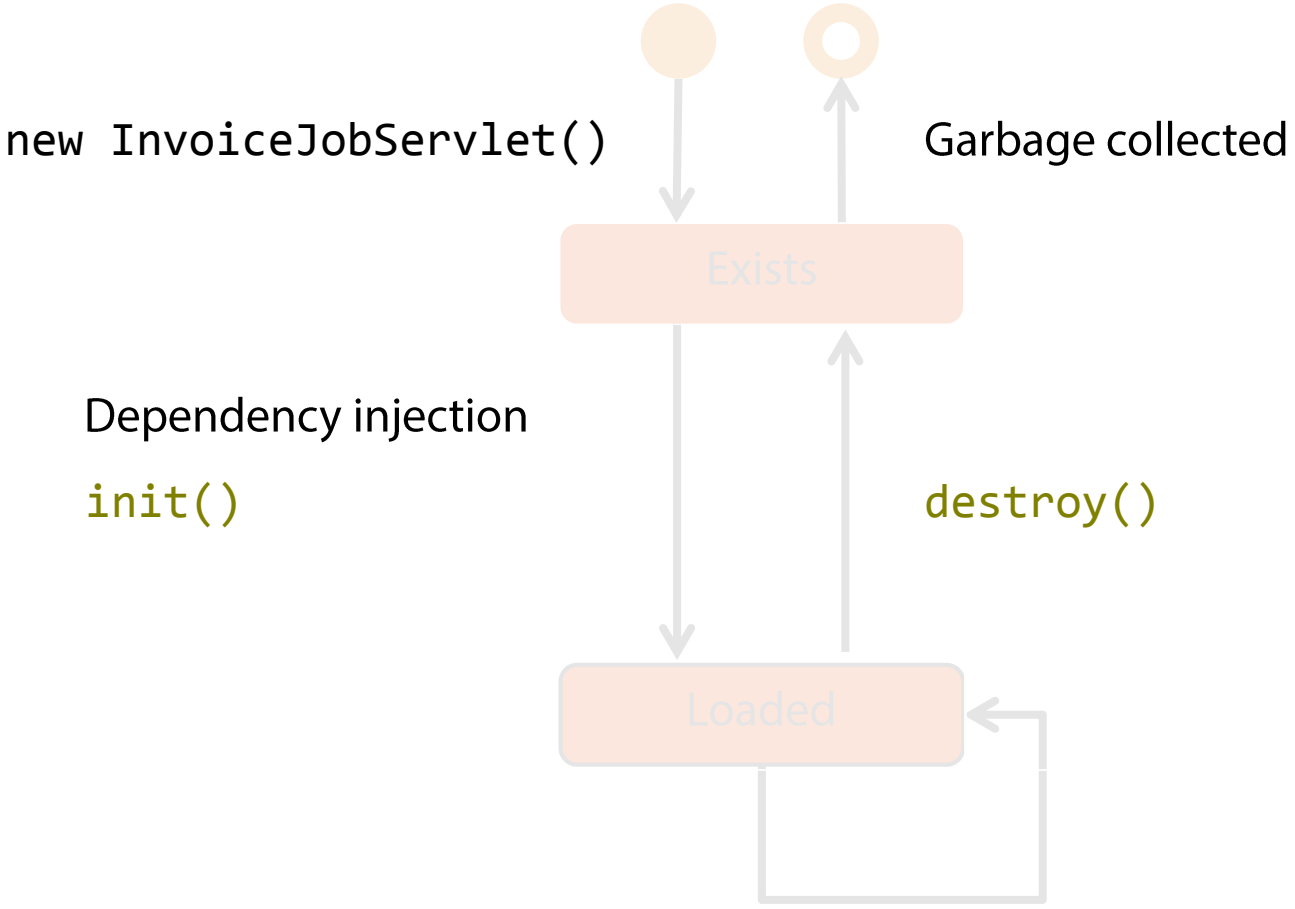


Servlet

```
@WebServlet(urlPatterns = "startJob")
public class InvoiceJobServlet extends HttpServlet

  @Override
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
                 throws ServletException, IOException {

    // business logic
  }

  @Override
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
                 throws ServletException, IOException {

    // business logic
  }
}
```

```
GET http://www.cdbookstore.com/startJob
```

Life Cycle of a Servlet



Deployment Descriptor

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                             http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">

  <display-name>cdbookstore</display-name>

  <session-config>
    <session-timeout>30</session-timeout>
  </session-config>
  <error-page>
    <error-code>404</error-code>
    <location>/error.html</location>
  </error-page>

</web-app>
```

web.xml

Servlets Packages

Package	Description
`javax.servlet`	Core Servlet API
`javax.servlet.annotation`	Servlet annotations
`javax.servlet.descriptor`	Deployment descriptor
`javax.servlet.http`	HTTP artifacts
`javax.servlet.resources`	XSD schemas for deployment descriptors

Processing a Request

- URL and parameters
- Dispatch the request
- Appropriate Servlet
- Multi-threaded
- Invokes business tier
- Handles the response

Handling Method

```
@WebServlet(urlPatterns = "invoice")
public class InvoiceServlet extends HttpServlet




  protected void doGet(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {









  }
}
```

```
GET http://www.cdbookstore.com/invoice
```

Handling a Request

```
@WebServlet(urlPatterns = "invoice")
public class InvoiceServlet extends HttpServlet




  protected void doGet(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {

    Long id = Long.valueOf(request.getParameter("id"));




  }
}
```

```
GET http://www.cdbookstore.com/invoice?id=1000
```

Invoking the Business Tier

```
@WebServlet(urlPatterns = "invoice")
public class InvoiceServlet extends HttpServlet

  @Inject
  private InvoiceService invoiceService;

  protected void doGet(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {

    Long id = Long.valueOf(request.getParameter("id"));

    Invoice invoice = invoiceService.findById(id);




  }
}
```

```
GET http://www.cdbookstore.com/invoice?id=1000
```

Handling the Response

```
@WebServlet(urlPatterns = "invoice")
public class InvoiceServlet extends HttpServlet

  @Inject
  private InvoiceService invoiceService;

  protected void doGet(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {

    Long id = Long.valueOf(request.getParameter("id"));

    Invoice invoice = invoiceService.findById(id);

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.print(invoice);
  }
}
```

```
GET http://www.cdbookstore.com/invoice?id=1000
```

Servlets

Invoice servlet

Get HTTP request

Invoke business tier

Display dynamic content

Web Pages

JavaServer Faces (JSF) 2.2

What Are Web Pages?

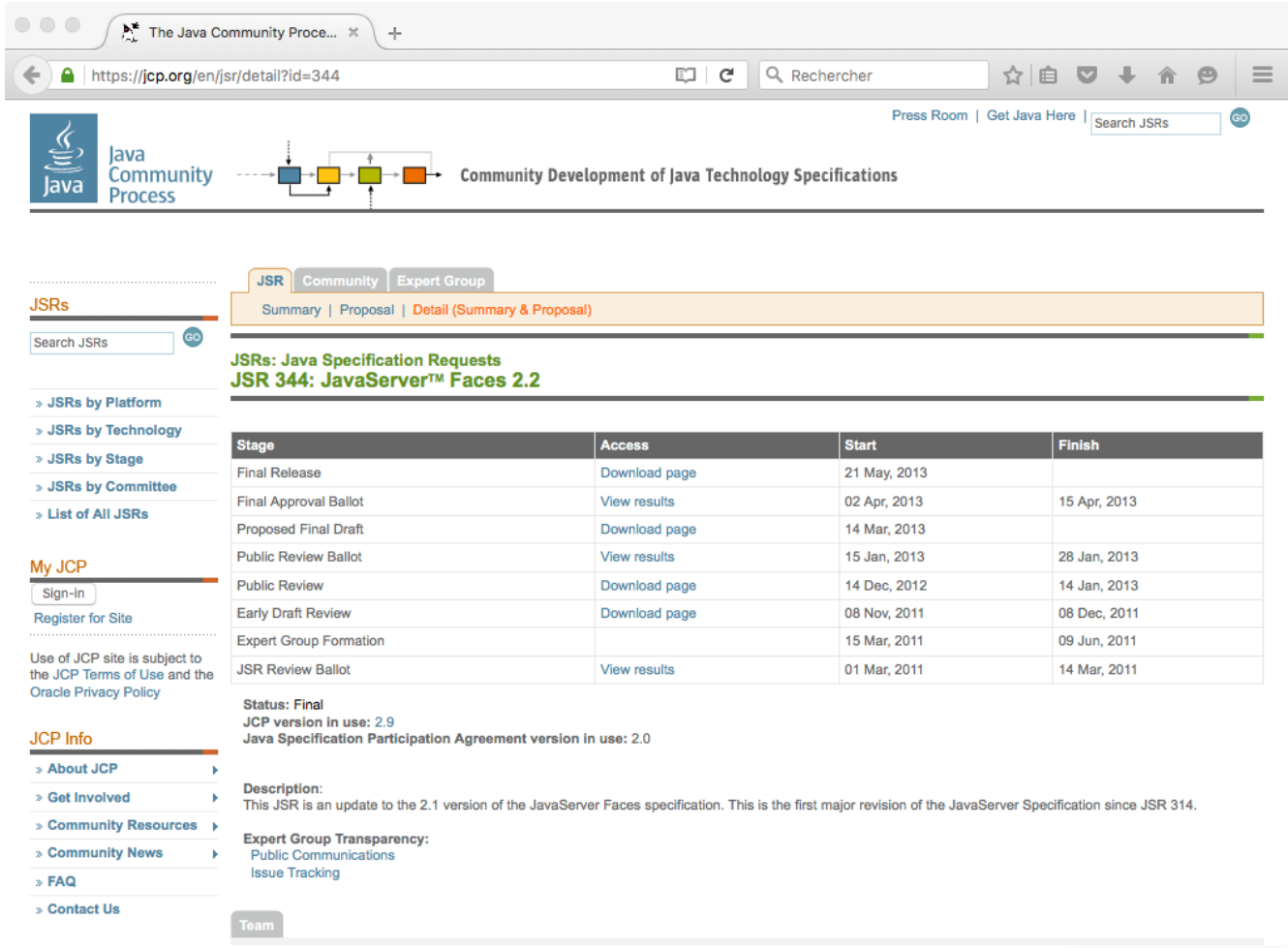
- Web user interface
- Dynamic content
- Static content
- Display on a browser
  - HTML
  - CSS
  - JavaScript

When to Use Web Pages

- Browsers are everywhere
- Web interfaces are richer
- Users expect more
- Social networks
- Web interfaces complex to develop

JSF Specification

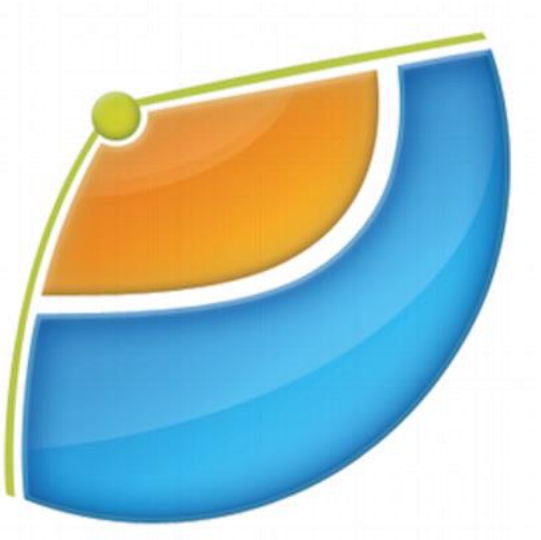
- JavaServer Faces 2.2
- JSR 344
- Expression Language 3.0
- JSR 341
- http://jcp.org/en/jsr/detail?id=344



Java Community Process

JSF Implementations

- Mojarra
- Apache MyFaces
- PrimeFaces
- IceFaces
- RichFaces



Web Pages

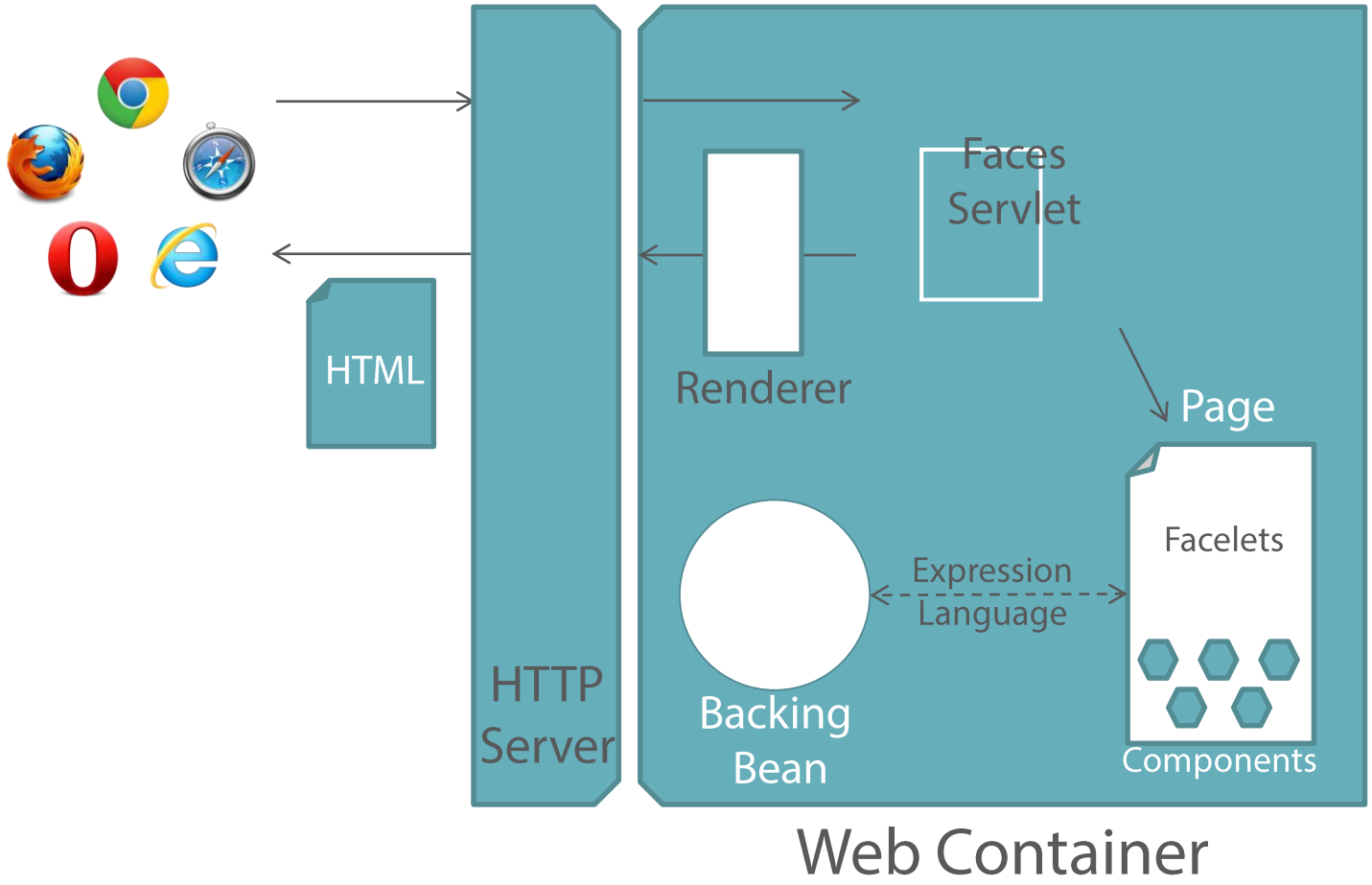
Admin

Catalog

User profile web pages

Understanding JSF

- MVC
- Faces Servlet
- Page
- Facelets
- Components
- Backing bean
- Expression Language
- Compiled to HTML



Page and Components

```
<f:view contentType="text/html" encoding="UTF-8">
  <h:head>
    <meta charset="utf-8"/>
  </h:head>
  <h:body>
    <h:form>
      <h:outputLabel   value="Login"/>
      <h:inputText     value="#{accountBean.login}"/>
      <h:outputLabel   value="Password"/>
      <p:password      value="#{accountBean.password}"/>
      <h:outputLabel   value="Date of birth"/>
      <p:calendar      value="#{accountBean.dateOfBirth}"/>

      <h:commandButton value="Sign up" action="#{accountBean.signup}"/>

    </h:form>
  </h:body>
</f:view>
```

Renderer

```
<f:view contentType="text/html" encoding="UTF-8">
 <h:head>
  <meta charset="utf-8"/>
 </h:head>

 <h:body>
  <h:form>
   <h:outputLabel value="Login"/>
   <h:inputText   value="#{accountBean.login}"/>

   <h:outputLabel value="Password"/>
   <p:password    value="#{accountBean.password}"/>

   <h:outputLabel value="Date of birth"/>
   <p:calendar    value="#{accountBean.dateOfBirth}"/>

   <h:commandButton value="Sign up"
                    action="#{accountBean.signup}"/>

  </h:form>
 </h:body>
</f:view>
```

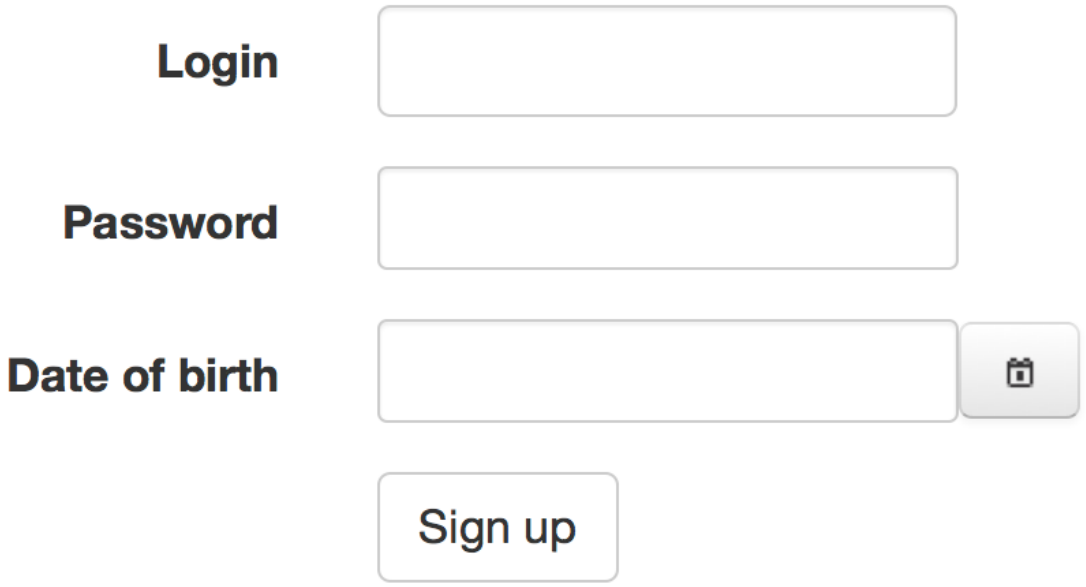
Facelets

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta charset="utf-8"/>
 </head>
 <body>
  <form method="post" action="/signup.xhtml">

   <label>Login</label>
   <input type="text"/>

   <label>Password</label>
   <input type="password"/>

   <label>Date of birth</label>
   <input type="text"/>

   <input type="submit" value="Sign up"/>

  </form>
 </body>
</html>
```

HTML

Renderer

```
<f:view contentType="text/html" encoding="UTF-8">
 <h:head>
  <meta charset="utf-8"/>
 </h:head>

 <h:body>
  <h:form>
   <h:outputLabel value="Login"/>
   <h:inputText   value="#{accountBean.login}"/>

   <h:outputLabel value="Password"/>
   <p:password    value="#{accountBean.password}"/>

   <h:outputLabel value="Date of birth"/>
   <p:calendar    value="#{accountBean.dateOfBirth}"/>

   <h:commandButton value="Sign up"
                    action="#{accountBean.signup}"/>

  </h:form>
 </h:body>
</f:view>
```



Expression Language

```
<f:view contentType="text/html" encoding="UTF-8">
  <h:head>
    <meta charset="utf-8"/>
  </h:head>
  <h:body>
    <h:form>
      <h:outputLabel   value="Login"/>
      <h:inputText     value="#{accountBean.login}"/>
      <h:outputLabel   value="Password"/>
      <p:password      value="#{accountBean.password}"/>
      <h:outputLabel   value="Date of birth"/>
      <p:calendar      value="#{accountBean.dateOfBirth}"/>

      <h:commandButton value="Sign up" action="#{accountBean.signup}"/>

    </h:form>
  </h:body>
</f:view>
```

Expression Language

- Print variables
- Access object attributes
- Invoke a method
- Operators

```
#{expression}
#{myVariable}
#{myObject.attribute}
#{myObject.doSomething}

#{2 < 3}

#{empty myObject}
```

Backing Bean

```
<f:view contentType="text/html" encoding="UTF-8">
 <h:head>
  <meta charset="utf-8"/>
 </h:head>

 <h:body>
  <h:form>
   <h:outputLabel value="Login"/>
   <h:inputText   value="#{accountBean.login}"/>

   <h:outputLabel value="Password"/>
   <p:password    value="#{accountBean.password}"/>

   <h:outputLabel value="Date of birth"/>
   <p:calendar    value="#{accountBean.dateOfBirth}"/>

   <h:commandButton value="Sign up"
                    action="#{accountBean.signup}"/>

  </h:form>
 </h:body>
</f:view>
```

```
@Named
public class AccountBean implements Serializable {

 private String login;
 private String password;
 private Date   dateOfBirth;

 public String signup() {

  if (em.createNamedQuery(FIND_BY_LOGIN, User.class)
     .setParameter("login", login)
     .getResultList().size() > 0) {

    return null;
  }

  user.setPassword(password);
  em.persist(user);
  return "/main";
 }
}
```

Deployment Descriptor

```
<faces-config xmlns="http://xmlns.jcp.org/xml/ns/javaee"
              xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
              http://xmlns.jcp.org/xml/ns/persistence/web-facesconfig_2_2.xsd"
              version="2.2">
  <application>
    <locale-config>
        <default-locale>en</default-locale>
    </locale-config>
  </application>

  <navigation-rule>
    <from-view-id>*</from-view-id>
    <navigation-case>
      <from-outcome>signup</from-outcome>
      <to-view-id>/account/signup.xhtml</to-view-id>
    </navigation-case>
  </navigation-rule>
</faces-config>
```

faces-config.xml

JSF Packages

Package	Description
`javax.faces`	Core JSF API
`javax.faces.component`	APIs for user interface components
`javax.faces.convert`	Classes and interfaces defining converters
`javax.faces.event`	Describing events and event listeners
`javax.faces.render`	Defining the rendering model
`javax.faces.webapp`	For integrating JSF into web applications

Writing JSF Pages

- JSF page different of HTML page
- JSF is a back-end technology
- JSF page needs to be rendered in HTML
- Combine JSF and HTML
- Component tree
- Tag libraries

JSF Page

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:p="http://primefaces.org/ui">

  <f:view contentType="text/html" encoding="UTF-8">
    <h:body>
      <h:form>
        <h:outputLabel    value="Login"/>
        <h:inputText      value="#{accountBean.login}"/>
        <h:outputLabel    value="Date of birth"/>
        <p:calendar       value="#{accountBean.dateOfBirth}"/>

        <h:commandButton value="Sign up" action="#{accountBean.signup}"/>
      </h:form>
    </h:body>
  </f:view>
</html>
```

JSF Page

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:p="http://primefaces.org/ui">

  <f:view contentType="text/html" encoding="UTF-8">
    <h:body>
      <h:form>
        <h:outputLabel   value="Login"/>
        <h:inputText     value="#{accountBean.login}"/>
        <h:outputLabel   value="Date of birth"/>
        <p:calendar      value="#{accountBean.dateOfBirth}"/>

        <h:commandButton value="Sign up" action="#{accountBean.signup}"/>
      </h:form>
    </h:body>
  </f:view>
</html>
```

Processing and Navigation

- Interact with a back-end system
- Navigate through other pages
- Backing beans
- Hold data
- CDI scopes
- Ajax calls natively

Backing Bean

```
@Named
@RequestScoped

public class AccountBean                                {

  private String login;









}
```

Backing Bean

```
@Named
@SessionScoped
@Transactional
public class AccountBean                                    {

  private String login;
  @Inject
  private EntityManager em;

  public String signup() {
    if (em.createNamedQuery(FIND_BY_LOGIN, User.class).setParameter("login", login)
        .getResultList().size() > 0) {

      return null;
    }
    return "/main";
 }
}
```

Method Call

```
<f:view contentType="text/html" encoding="UTF-8">
  <h:head>
    <meta charset="utf-8"/>
  </h:head>
  <h:body>
    <h:form>
      <h:outputLabel   value="Login"/>
      <h:inputText     value="#{accountBean.login}"/>
      <h:outputLabel   value="Date of birth"/>
      <p:calendar      value="#{accountBean.dateOfBirth}"/>

      <h:commandButton value="Sign up" action="#{accountBean.signup}">

      </h:commandButton>
    </h:form>
    <h:outputLabel   value="Login status" id="status"/>
  </h:body>
</f:view>
```

Ajax Method Call

```
<f:view contentType="text/html" encoding="UTF-8">
  <h:head>
    <meta charset="utf-8"/>
  </h:head>
  <h:body>
    <h:form>
      <h:outputLabel    value="Login"/>
      <h:inputText      value="#{accountBean.login}"/>
      <h:outputLabel    value="Date of birth"/>
      <p:calendar       value="#{accountBean.dateOfBirth}"/>

      <h:commandButton value="Sign up" action="#{accountBean.signup}">
        <f:ajax execute="@form" render=":status"/>
      </h:commandButton>
    </h:form>
    <h:outputLabel   value="Login status" id="status"/>
  </h:body>
</f:view>
```

Web Pages

JSF page

Backing bean

Expression language

Web Sockets

WebSockets 1.0

What Are Web Sockets?

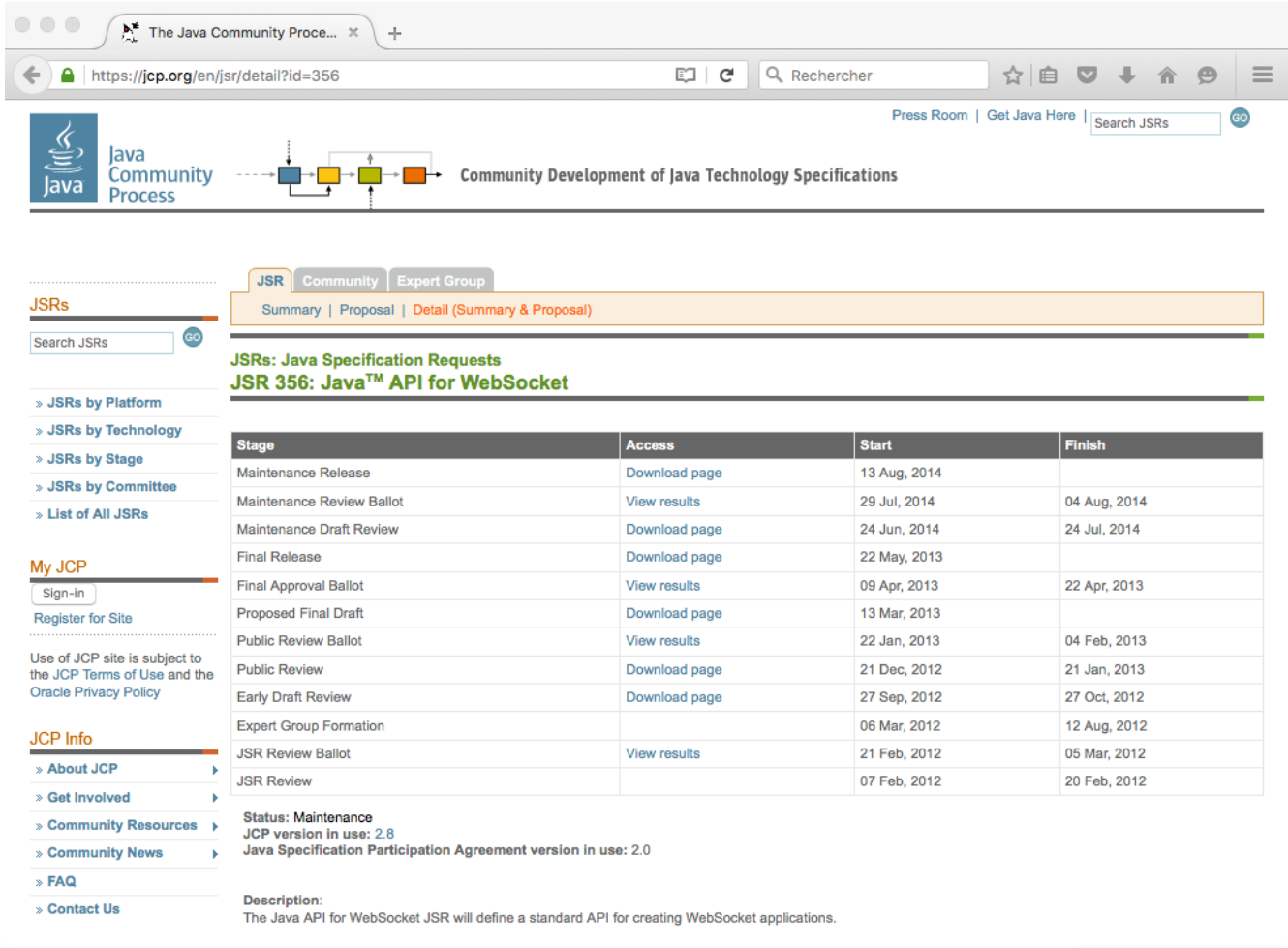
- Ease communication between client and server
- Full-duplex communication
- Text and binary messages
- Without the HTTP request/response lifecycle
- Send a message at any time
- Asynchronous data delivery

When to Use Web Sockets

- Multiple users communicating with each other
- Data constantly changing
- Social application
- Multiplayer games
- Financial dashboard

WebSocket Specification

- WebSocket 1.0
- JSR 356
- http://jcp.org/en/jsr/detail?id=356



Java Community Process

WebSocket Implementations

- Tyrus
- Tomcat
- Jetty
- Undertow

jetty://

undertow

Web Sockets

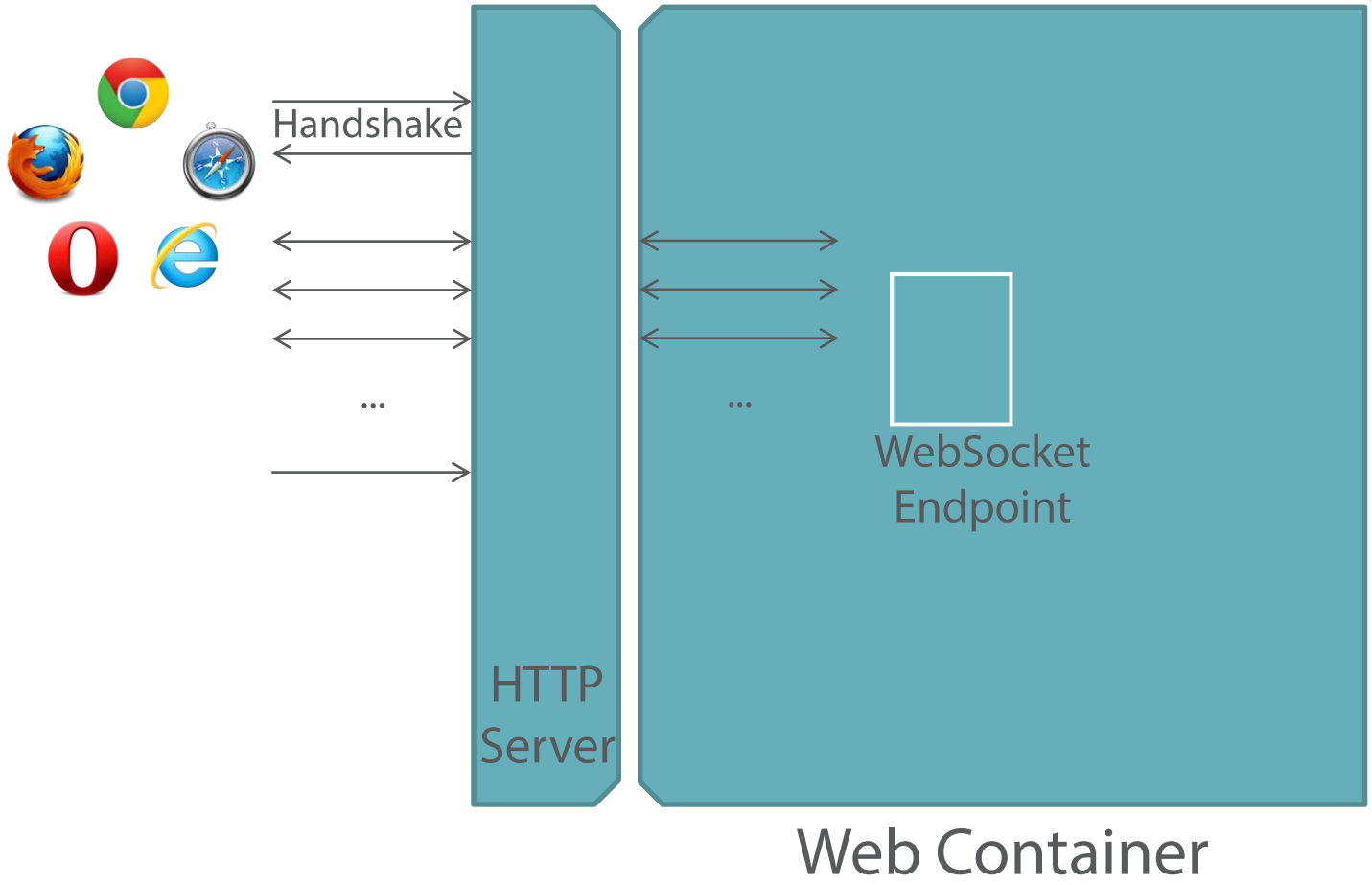
Chat

Data updated frequently

Multiple clients

Understanding WebSockets

- HTTP upgrades
- Handshake
- Data framing
- WebSocket Endpoint
- Stays open
- Pushes information
- Closes connection



WebSocket Endpoint

```
@ServerEndpoint("/chat")
public class ChatEndpoint {

}
```

http://www.myhost.com/**chat**

WebSocket Endpoint

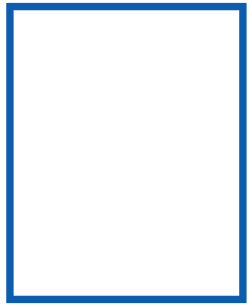
```
@ServerEndpoint("/chat")
public class ChatEndpoint {

  @OnOpen
  public void onOpen(Session session) { // ... }

  @OnMessage
  public void message(String message, Session client) throws Exception {
    for (Session peer : client.getOpenSessions()) {
      peer.getBasicRemote().sendText(message);
    }
  }

  @OnClose
  public void onClose(Session session) { // ... }

}
```

```
ws://www.myhost.com/chat
```

WebSocket Packages

Package	Description
`javax.websocket`	Core WebSocket API
`javax.websocket.server`	Server endpoint

Summary



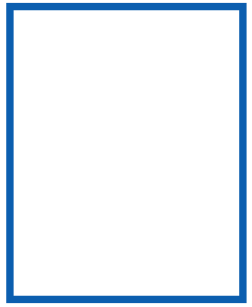
Web tier

Servlets

JSF Pages

WebSockets

What's Next



Interoperability tier

XML & JSon

JAX-RS

JMS