CS544 EA

Applications

Web Containers

Web Containers

- We’ll first discuss :
  - Web Container IoC
  - Servlets (beans)
  - Filters (interceptors)
- Then we’ll look at JSP for view:
  - JSPs are all XML
  - JSTL tags for program control
  - EL inside statements and to print

Containers so Far

- We saw that Spring is a container:
  - IoC (creates objects)
  - DI (connects them)
  - AOP (proxies for extra functionality)

- Hibernate is also a container:
  - Creates Objects (IoC)
  - Connects objects based on associations (DI)
  - Proxies to provide lazy loading (AOP)

Web Container

- We will see that a web container:
  - Creates Objects (IoC)
  - Can add proxies for extra functionality (Filters)
  - Does not connect objects together (no DI)

- Main difference, web containers work with:
  - Incoming Request objects
  - Outgoing Response objects

Not POJOs

- Another big difference is that the objects managed by web containers are not POJOs
  - To be a Servlet or Filter you have to extend or implement a Technology related class / interface
  - Web containers design is old
  - Before Rod Johnson’s book about POJO containers

Comparing Terminology

- Servlet:
  - Object that the container creates and manages
  - What Spring called a Bean
- Web.xml
  - Configuration file that configures the container
  - What spring called springconfig.xml
- Filter
  - Proxy for a Servlet
  - Somewhat similar to an @Around advice

Web.xml

Inside project's /WEB-INF/

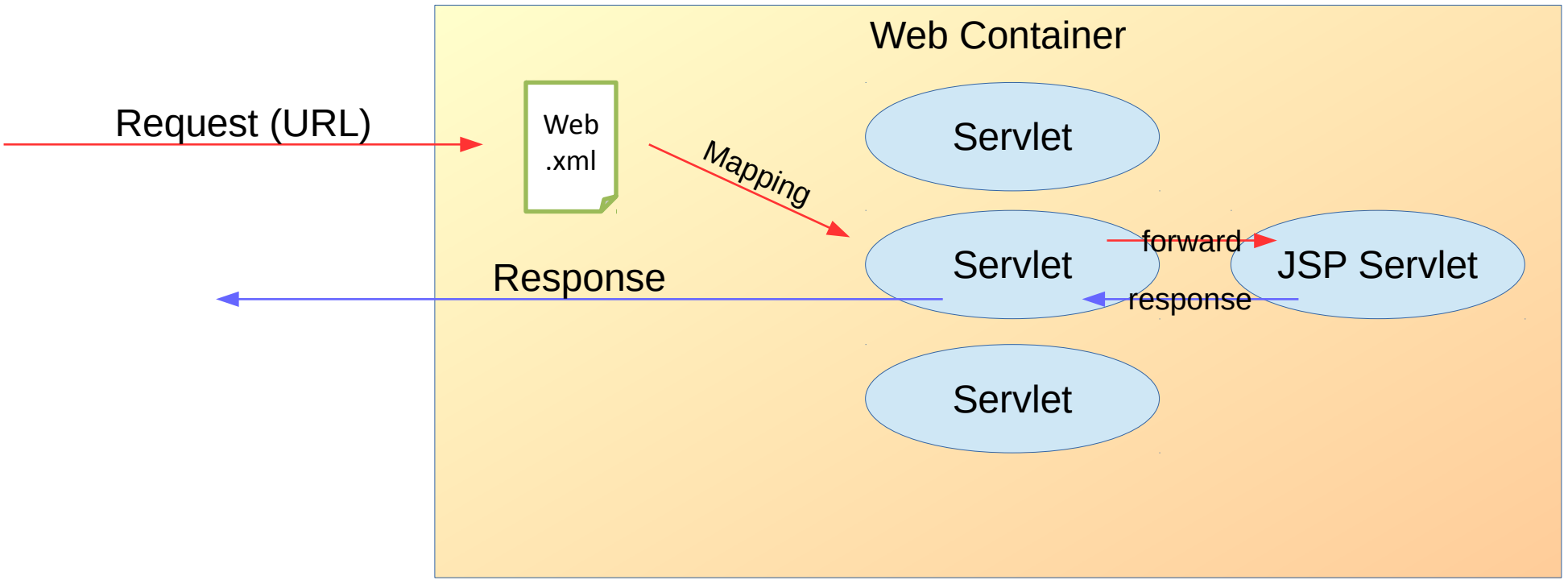
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

  <servlet>
    <servlet-name>Servlet Demo</servlet-name>
    <servlet-class>demo.ServletDemo</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>Servlet Demo</servlet-name>
    <url-pattern>/servlet</url-pattern>
  </servlet-mapping>
</web-app>
```

Can also be done with @WebServlet

Request / Response

- The container receives a request for a URL
  - Looks at Servlet-Mappings to find a matching pattern
  - Passes request and a empty response to servlet
    - Request may contain additional key/value params
  - Servlet reads request, and fills in response
    - Optionally forwarding req/resp to other servlet for more
  - Response (text output) then printed to user

Visually


Web Container
Request (URL)
Web
.xml
Mapping
Servlet
Servlet
forward
JSP Servlet
Response
response
Servlet

Requests

- HTTP (web) Requests have a type:
  - GET or POST for HTML
- May have key / value pair parameters:
  - GET in URL, POST as ‘post data’
- Often also a Session ID cookie
  - Allows the server to find storage for this user

Servlet

```java
@WebServlet(name = "Hello", urlPatterns = { "/Hello" })
public class Hello extends HttpServlet {
	private static final long serialVersionUID = 1L;

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

		request.setAttribute("now", new Date());
		request.setAttribute("one", 1);
		request.setAttribute("two", 2);

		ServletContext context = this.getServletContext();
		String jsp = "/Hello.jsp";
		RequestDispatcher dispatcher = context.getRequestDispatcher(jsp);
		dispatcher.forward(request, response);
	}
}
```

Not a POJO
Extends HttpServlet

Method on HttpServlet
Lets you get container
(Context)

Filter

```
@WebFilter(filterName = "OpenEntityManagerInView", urlPatterns = "/*")
public class EntityManagerInterceptor implements Filter {
     @Override
     public void destroy() { }
     @Override
     public void init(FilterConfig fc) throws ServletException { }
     @Override
     public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
               throws IOException, ServletException {

          EntityManager em = EntityManagerHelper.getCurrent();
          try {
               em.getTransaction().begin();
               chain.doFilter(req, res);
               em.getTransaction().commit();
          } catch (RuntimeException e) {

               if (em != null && em.isOpen())
                    em.getTransaction().rollback();
               throw e;

          } finally {
               em.close();
          }
     }
}
```

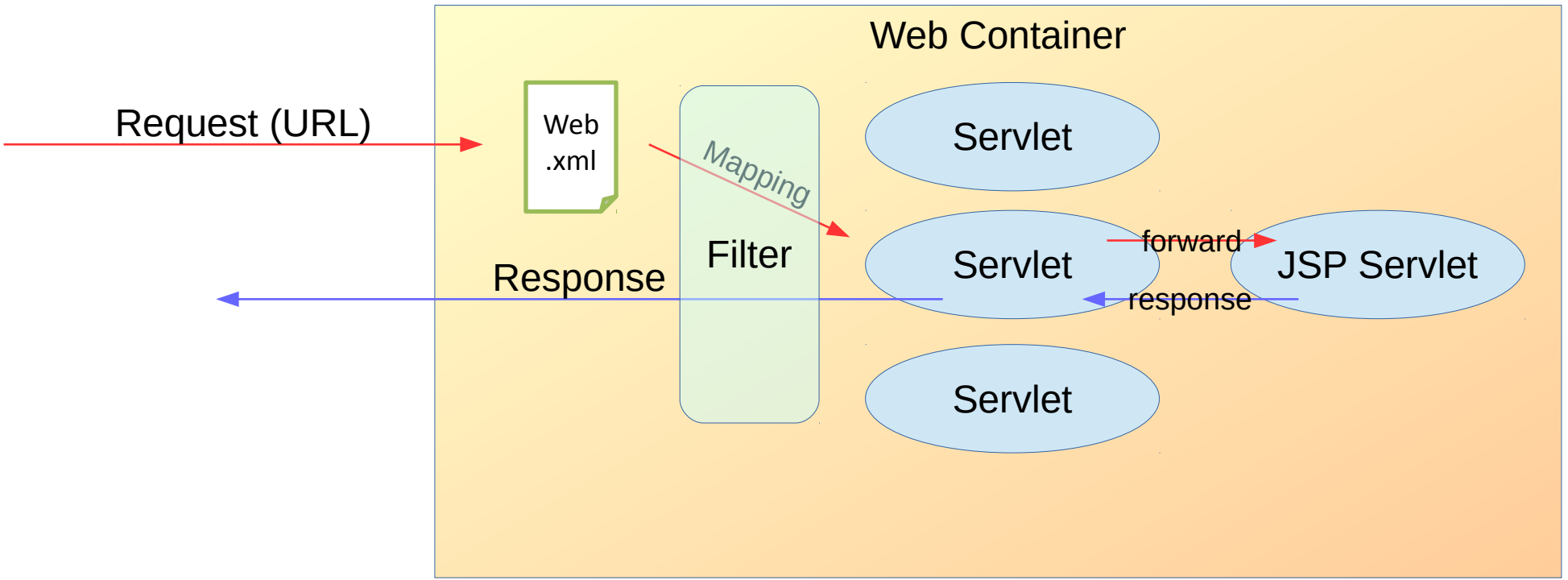
Not a POJO
Implements Filter

Web.xml for Filter

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

  ...

  <filter>
    <filter-name>OpenEntityManagerInView</filter-name>
    <filter-class>example.filter.OpenEntityManagerInView</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>OpenEntityManagerInView</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  ...


</web-app>
```

Can also be done with @WebFilter

Visually


Web Container
Request (URL)
Web
.xml
Mapping
Filter
Servlet
Servlet
forward
JSP Servlet
response
Response
Servlet