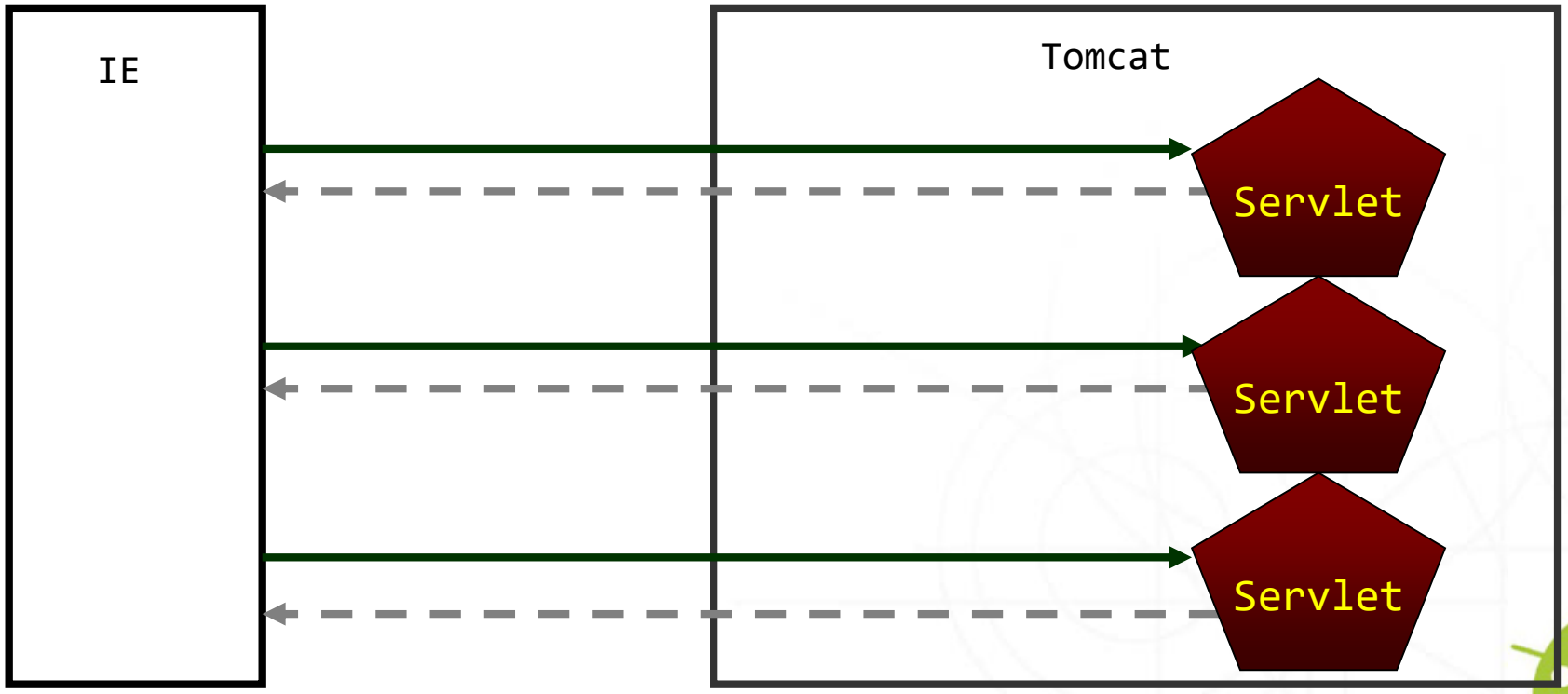
Servlet Filter 過濾器

段維瀚 老師

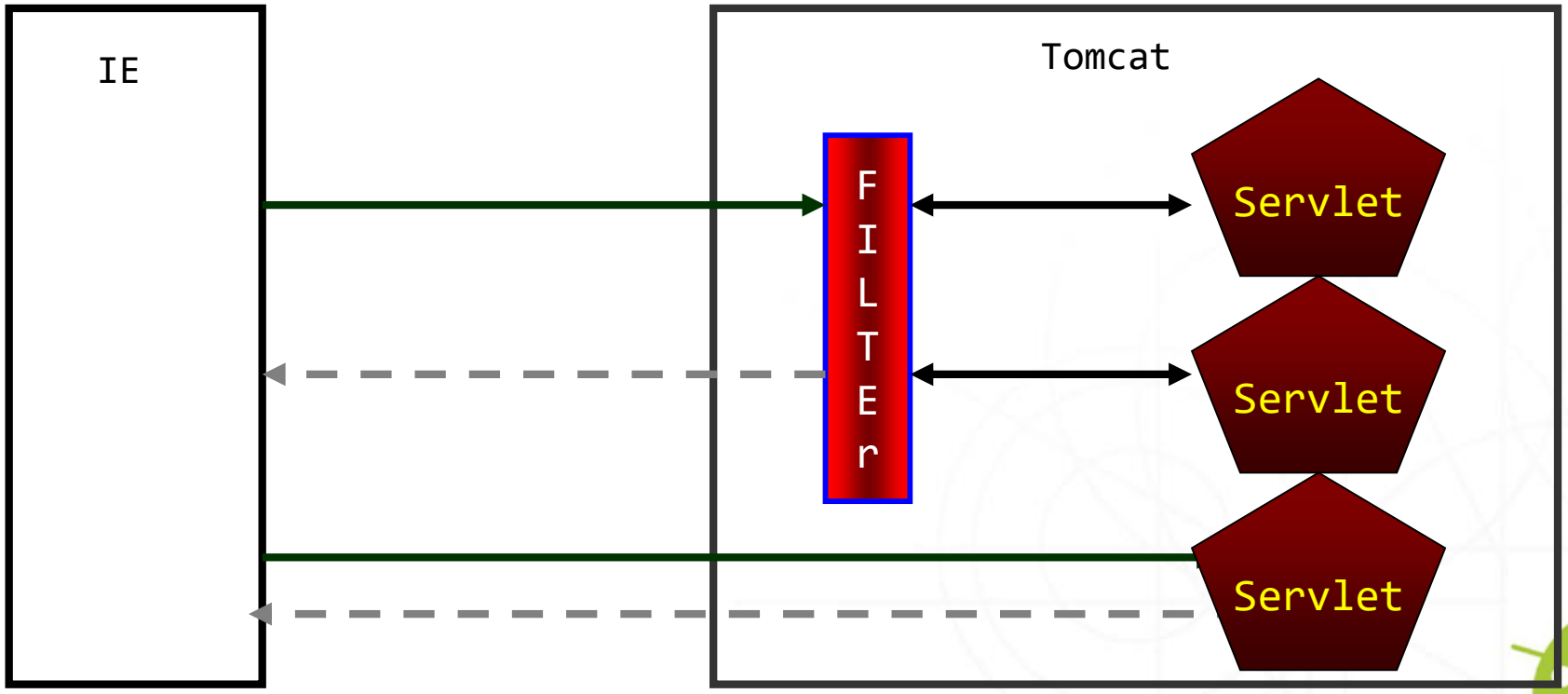
Filter

- Filter 過濾器
  - 可以擔任瀏覽器與JSP/Servlet之間的一個中介處理者，一些request的前置處理動作及一些response的後置處理，都可以交由這個中介處理者來完成
  - 例如某些網頁都需要統一的身份驗證方式時，與其在每一個網頁中都撰寫驗證的程式碼，不如直接撰寫Filter，讓它來統一進行處理。

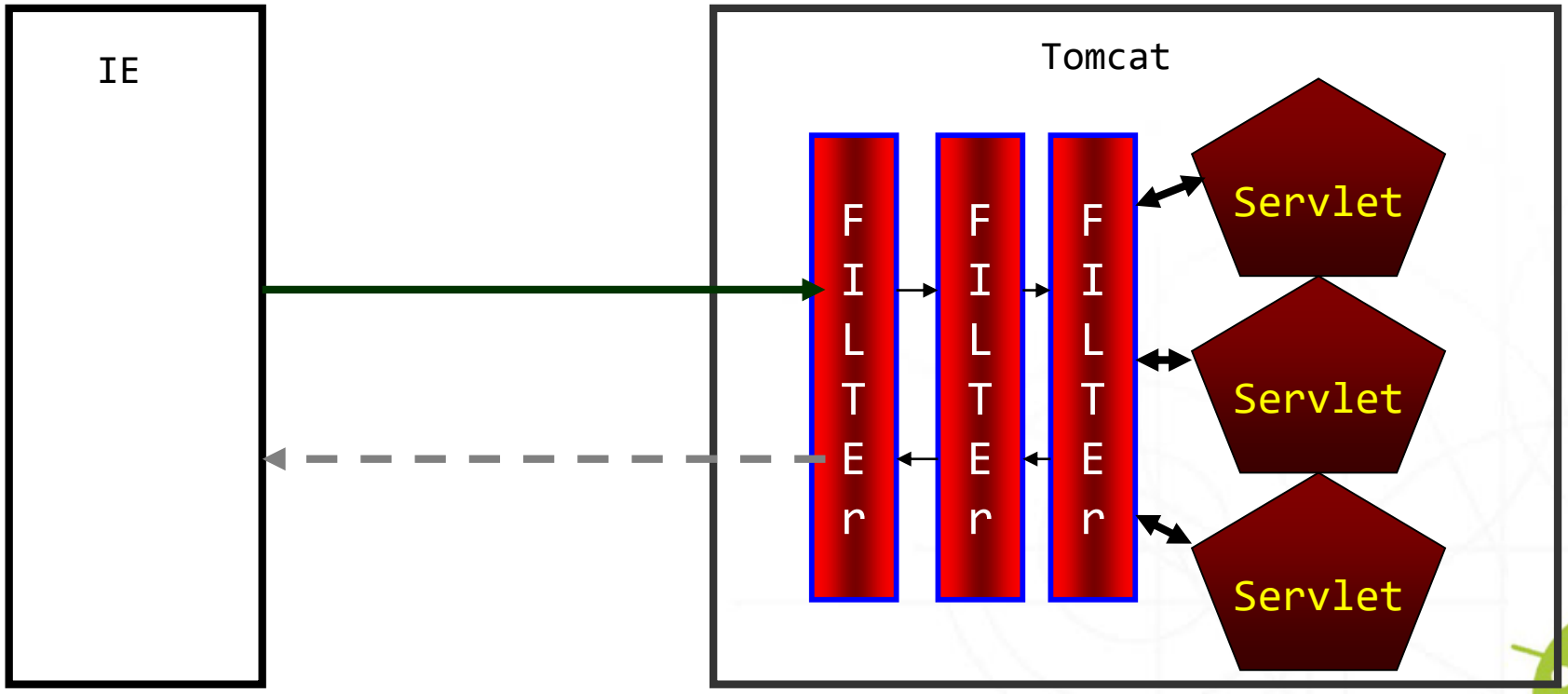
Filter 運作



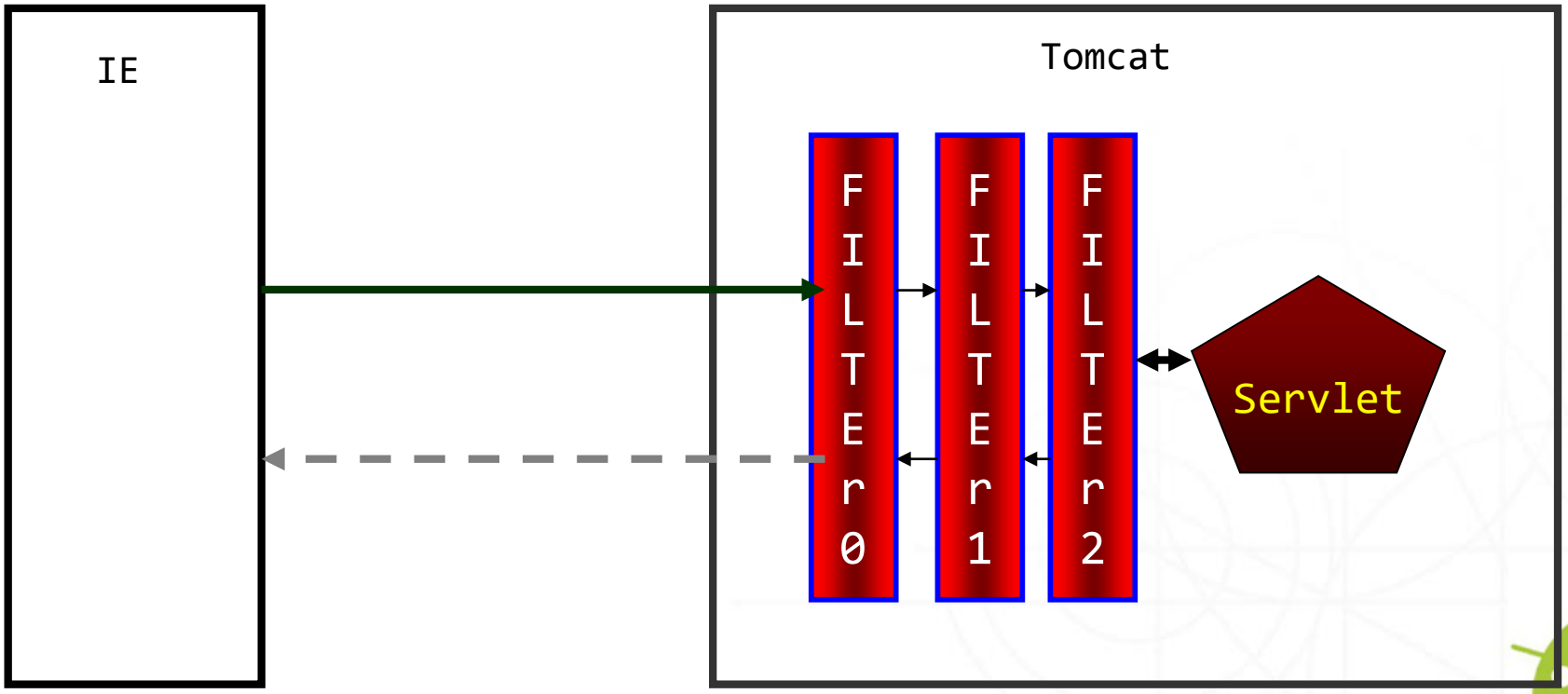
Filter 運作



Filter 運作

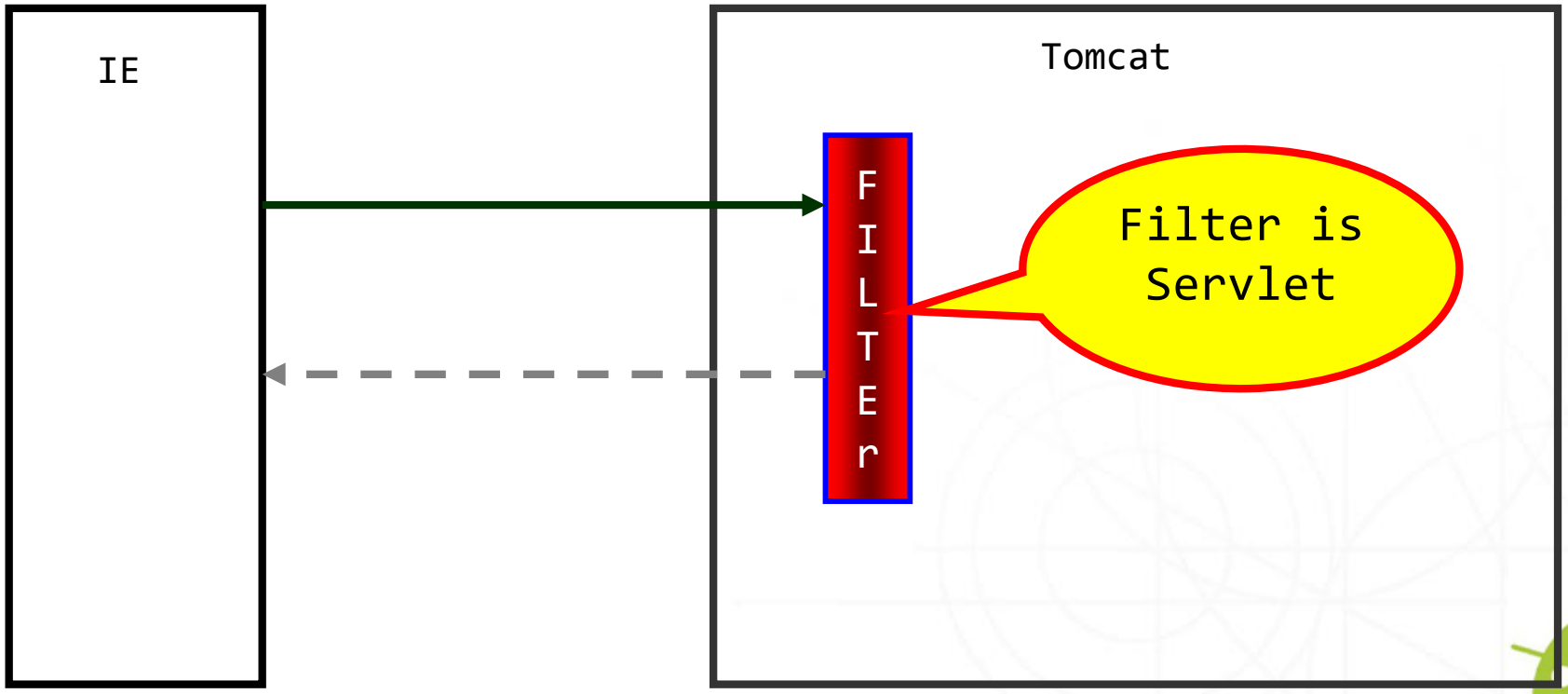


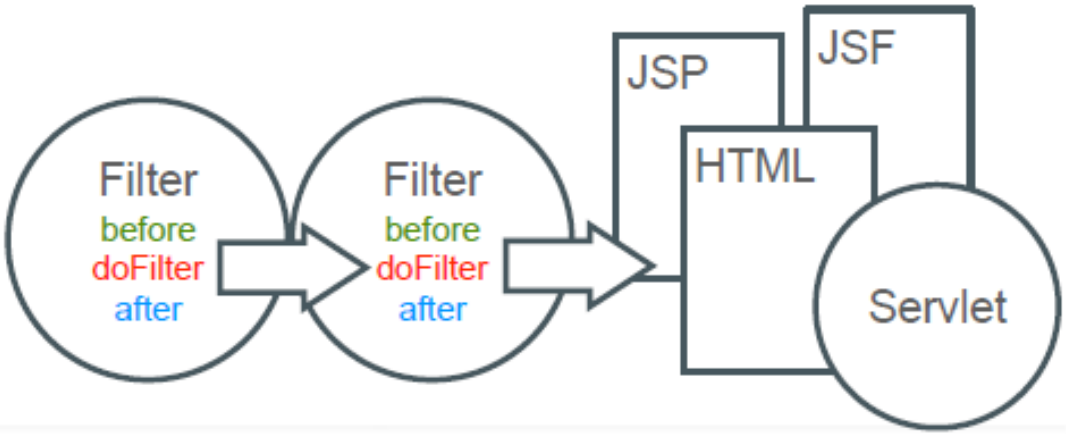
執行順序



1：根據 web.xml 中控制 filter 的位置前後來控制順序
2：若使用 @WebFilter 則是根據 Filter 的 Java 類別名稱字母大小來決定執行順序
Filter0.java -> Filter1.java -> Filter2.java

Filter 運作






```
@WebFilter(filterName = "SomeFilter", urlPatterns = {"/*"})
public class SomeFilter implements Filter {
public void doFilter(ServletRequest request,
                     ServletResponse response,
                     FilterChain chain)
                  throws IOException, ServletException {
   // Perform "Before" Actions
      try {
         chain.doFilter(request, response);
      }catch (Throwable t) {
      // Perform "After Error" Actions
      }
   // Perform "After" Actions
   }
}
```

Filter

- Filter 過濾器實作
  - Filter實際上是一個純粹的Java類別程式，它要實作javax.servlet.Filter介面，這個介面中有三個實作的方法：
    - init()、destory()與doFilter()。
      - init()是Filter類別被載入時會執行的方法
      - doFilter()則是實作Filter功能的核心，您想要Filter完成的工作就撰寫在其中
      - destory()是 Filter物件生命週期結束時會執行的方

Filter

- Filter
  - 過濾器介面實作
    - init()是Filter類別被載入時會執行的方法
    - destory()是 Filter物件生命週期結束時會執行的方
    - doFilter()則是實作Filter功能的核心，您想要Filter完成的工作就撰寫在其中
  - GenericFilter
  - HttpFilter

Filter

- Filter 過濾器實作
  - init()是Filter類別被載入時會執行的方法
  - destory()是 Filter物件生命週期結束時會執行的方
  - doFilter()則是實作Filter功能的核心，您想要Filter完成的工作就撰寫在其中

Filter

Servlet 3.0
Tomcat 8

```
public interface Filter {
    public void init(FilterConfig filterConfig)
                                            throws ServletException;
    public void doFilter(ServletRequest sr, ServletResponse sr1,
                         FilterChain fc) throws IOException, ServletException;
    public void destroy();
}
```

Servlet 4.0
Tomcat 9, TomEE 8

```
public interface Filter {
    public default void init(FilterConfig filterConfig)
                                            throws ServletException {}
    public void doFilter(ServletRequest sr, ServletResponse sr1,
                         FilterChain fc) throws IOException, ServletException;
    public default void destroy() {}
}
```

HttpFilter (Servlet 4.0)

```
public abstract class HttpFilter extends GenericFilter {

    public HttpFilter() {
    }

    public void doFilter(ServletRequest req, ServletResponse res,
                         FilterChain chain)
                         throws IOException, ServletException {


    }

    protected void doFilter(HttpServletRequest req, HttpServletResponse res,
                            FilterChain chain)
                            throws IOException, ServletException {


    }
}
```

HttpFilter

- Servlet 4.0 pom.xml 配置

```
<dependency>
    <groupId>javax</groupId>
    <artifactId>javaee-web-api</artifactId>
    <version>8.0.1</version>
    <scope>provided</scope>
</dependency>
```

量測執行效能過濾器

```
@WebFilter("/servlet/*")
public class PerformanceFilter extends HttpFilter {

  @Override
  protected void doFilter(HttpServletRequest request,
                          HttpServletResponse response, FilterChain chain)
                          throws IOException, ServletException {
    long begin = System.currentTimeMillis();
    chain.doFilter(request, response);
    long end = System.currentTimeMillis();
    System.out.println( chain.getClass() + " : " + (begin - begin) + " ms");
  }
}
```

@Filter

- @WebFilter("/*")
- @WebFilter(servletNames={"SomeServlet"})
-
```
@WebFilter(
    urlPatterns={"/*"},
    initParams={
        @WebInitParam(name = "PARAM1", value = "VALUE1"),
        @WebInitParam(name = "PARAM2", value = "VALUE2")
    })
```
- @WebFiler 調用順序是按照 name 的 ASCII來排序

web.xml

- Filter 過濾器實作(web.xml)-part I
  - <filter>
    <filter-name>FilterA</filter-name>
    <filter-class>filters.FilterA</filter-class>
    </filter>
  - <filter-mapping>
    <filter-name>FilterA</filter-name>
    <url-pattern>/*</url-pattern>
    </filter-mapping>

web.xml

- Filter 過濾器實作(web.xml)-part II
  - <filter>
    <filter-name>FilterA</filter-name>
    <filter-class>filters.FilterA</filter-class>
    </filter>
  - <filter-mapping>
    <filter-name>FilterA</filter-name>
    <servlet-name>RedServlet</servlet-name>
    </filter-mapping>

web.xml

- Filter 過濾器實作(web.xml)-part III
  - ```
    <filter>
      <filter-name>FilterA</filter-name>
      <filter-class>filters.FilterA</filter-class>
      <init-param>
        <param-name>PARAM1</param-name>
        <param-value>VALUE1</param-value>
      </init-param>
      <init-param>
        <param-name>PARAM2</param-name>
        <param-value>VALUE2</param-value>
      </init-param>
    </filter>
    <filter-mapping>
      <filter-name>FilterA</filter-name>
      <url-pattern>/*</url-pattern>
    </filter-mapping>
    ```


    ```

web.xml

- Filter 過濾器實作(web.xml)-part VI
  - <filter>
    <filter-name>FilterA</filter-name>
    <filter-class>filters.FilterA</filter-class>
    </filter>
  - <filter-mapping>
    <filter-name>FilterA</filter-name>
    <url-pattern>/*</url-pattern>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>INCLUDE</dispatcher>
    <dispatcher>ERROR</dispatcher>
    </filter-mapping>

Filter

- extends HttpServletRequestWrapper
  - 繼承 HttpServletRequestWrapper 讓Filter可以容易取得請求資料，必要時可加以變更。
- extends HttpServletResponseWrapper
  - 繼承 HttpServletResponseWrapper 讓Filter可以容易取得 out 的內容物。用以達成變更 HTTP 相關資料內容

MyRequest.java

```
public class MyRequest extends HttpServletRequestWrapper {
    private Map<String, String> params = new HashMap<>();
    public MyRequest(HttpServletRequest request) { // 一定要寫
        super(request);
    }
    public String getParameter(String name) {
        String value = params.get(name);
        if(value == null) {
            value = super.getParameter(name);
        }
        return value;
    }
    public void setParameter(String name, String value) {
        params.put(name, value);
    }
}
```

ExchangeFilter.java

```
@WebFilter("/servlet/exchange/price")
public class ExchangeFilter extends HttpFilter {
    @Override
    protected void doFilter(HttpServletRequest req,
                            HttpServletResponse res,
                            FilterChain chain)
                            throws IOException, ServletException {

        MyRequest myRequest = new MyRequest(req);
        myRequest.setParameter("value", req.getParameter("name") + "0");
        chain.doFilter(myRequest, res);

    }
}
```

MyResponse.java

```
public class MyResponse extends HttpServletResponseWrapper {

    private PrintWriter out;
    private CharArrayWriter bufferedWriter;

    public MyResponse(HttpServletResponse response) { // 一定要寫
        super(response);
        bufferedWriter = new CharArrayWriter();
        out = new PrintWriter(bufferedWriter);
    }

    @Override
    public PrintWriter getWriter() {
        return out;
    }

    public String getResult() {
        return bufferedWriter.toString();
    }
}
```

WatermarkFilter.java

```
@WebFilter("/report/*")
public class WatermarkFilter implements Filter {

   @Override
   public void doFilter(ServletRequest req, ServletResponse resp,
                        FilterChain chain)
                        throws IOException, ServletException {

      MyResponse myResp = new MyResponse((HttpServletResponse)resp);
      chain.doFilter(req, myResp);
      String html = myResponse.getResult();
      html = html.replaceAll("<body",
                  "<body background=\"../images/watermark.jpg\"");
      resp.getWriter().println(html);
   }

}
```


JUnit
JS
ORACLE
FUSION MIDDLEWARE
WEBLOGIC SERVER
JBoss
by Red Hat
HIBERNATE
HTML
ORACLE
SPATIAL
PostGIS
SoapUI
by SMARTBEAR
spring
by Pivotal
GeoServer
CSS
eclipse link
OpenLayers
SUBVERSION
jQuery
write less, do more.
PRIME FACES
Java™
Microsoft
SQL Server
mongoDB
NetBeans
maven
git
MySQL
ORACLE
DATABASE
ANGULARJS
by Google
LIFERAY
eclipse
PostgreSQL
sonarqube
Visual Studio