**Filter in java**

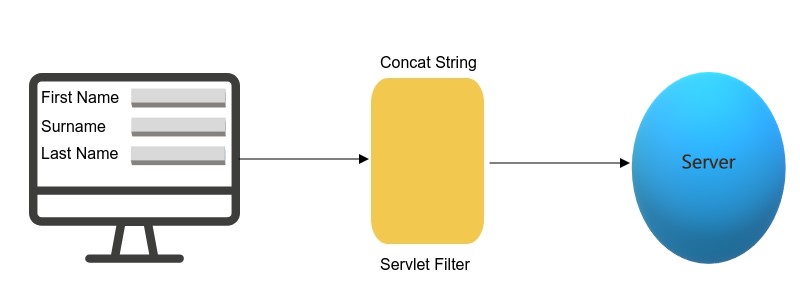
**Introduction**

Filter is not servlet. Filter is something we do behind scene, they work behind servlet. Filter is not front end or presentation layer task but it is administrator task. Filter is preprocessors of request and postprocessors of response. Filter are simply treatment of data which allow modification, alteration, encryption, decryption and compression of data.

**Where does we use filters?**

In application, if we are sending large image to server than it will take more time as image must be posted and after that it should be received by sever. Once we request page then it has to go to target in between we can not stop it anywhere. Filter are provision which allow to do something even after submitting form. We can use filter to compress image between image is posted and received by server. Filter are used between request and till request reaches to target.

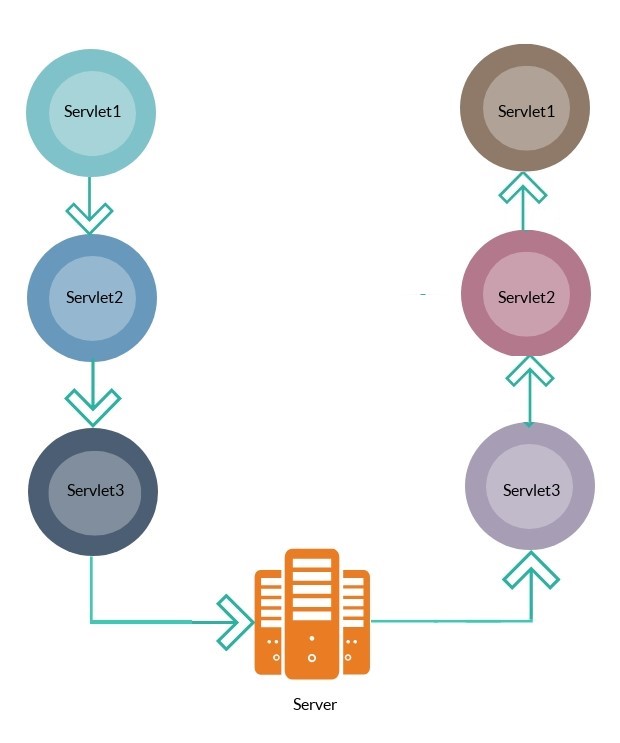
In application we are asking for last name, first name and surname field as user don’t make any mistake but at server side script we are having only name field. So we wish that fully concated string must go to server. We can use filter to concat last name, first name and surname to single string.

****

We use servlet for writing admistrative code. We should never write login code in servlet because if that script is leaked then anyone can know how we are checking password and they can easily use SQL injection to leak your whole code.

**How filter work?**

Filter never goes to end user. User can call servlet but user can not call filter. Nobody can directly use filter and it work in background. We can also have multiple filter, one filter for login, other for compression of image, etc. When you go to airport there are many security gates, somebody will check your passport, somebody will check your bags same way consider all of them as filter before you go to your servlet.



Instead of writing sensitive code in core business servlet we should write it into filter because no one is going to execute that filter except to whom permission is given. Developer or admistrator of servlet can configure filter for particular servlet.

**Type of filter**

We have two ways of filtering:

1. Filtering of request.

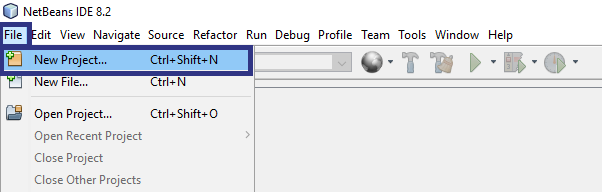
Now if we want to note down IP address of every user who visit my site then we can write filter which will execute before servlet is executed. We don’t write this kind of code in servlet because this task is admistrative task.

1. Filtering of response.

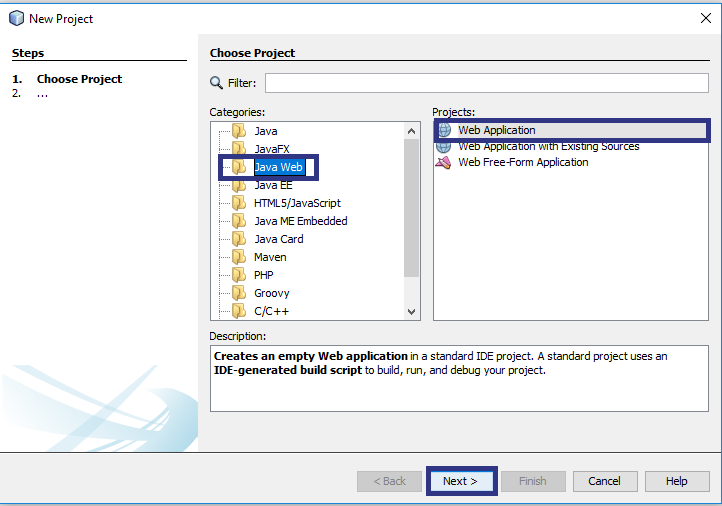
Sometimes we want to treat response data also. Server has given different response but user is getting different response treated by filter. Filter is kind of treatment on data in request and response.

**Example of filter**

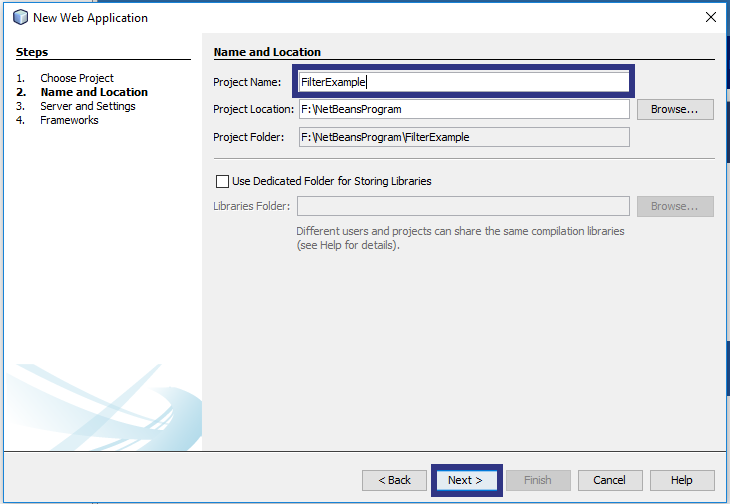
**Step: 1**



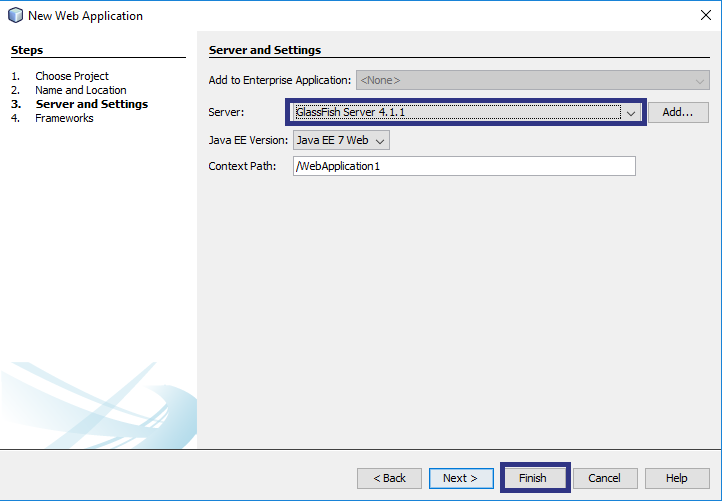
**Step: 2**



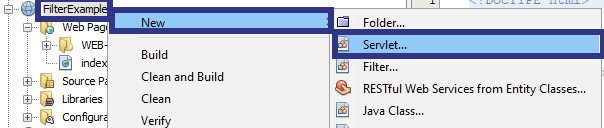
**Step:3**

****

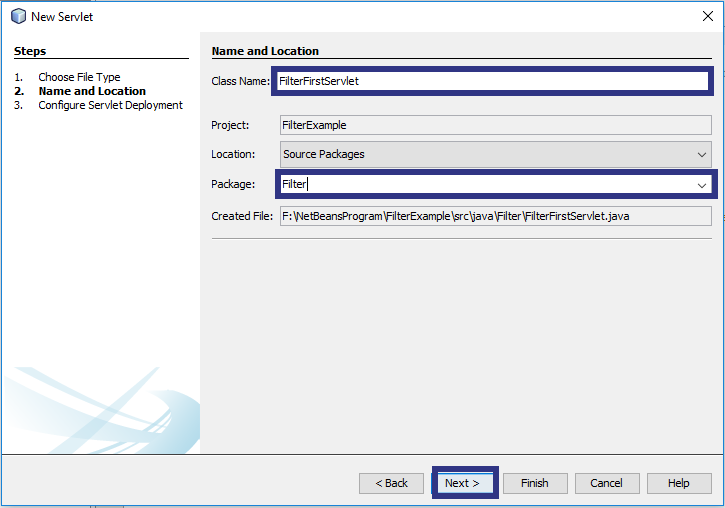
**Step: 4**

****

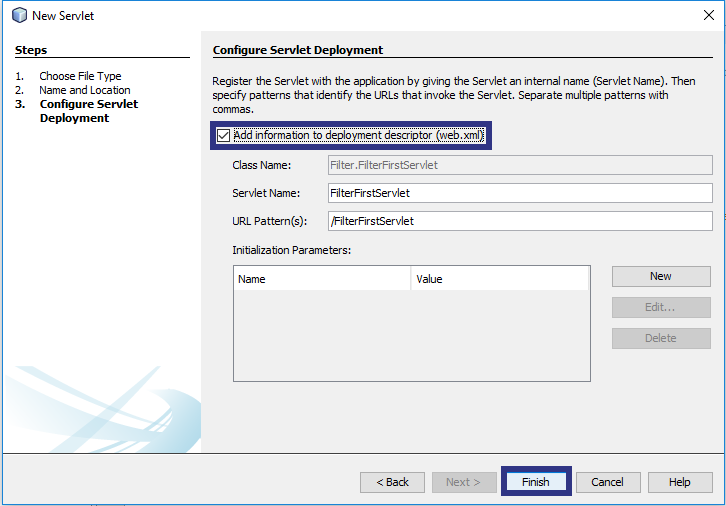
**Step: 5**

****

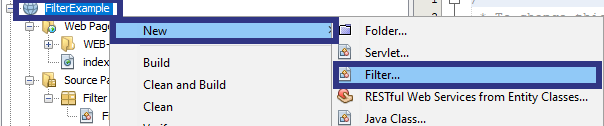
**Step: 6**

****

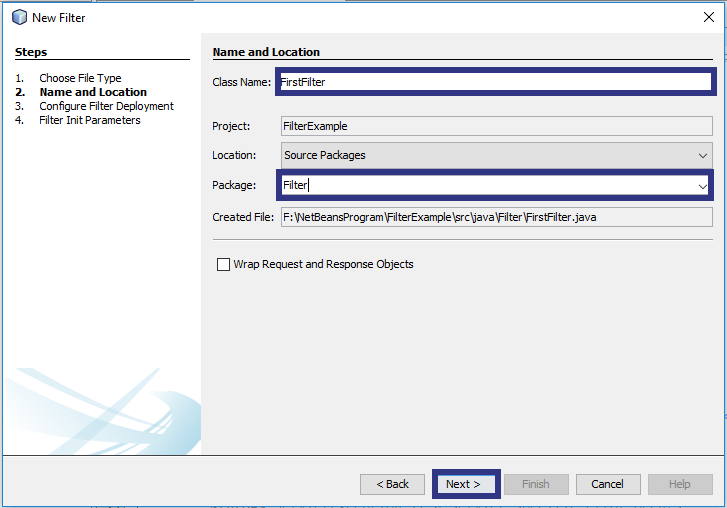
**Step: 7**

****

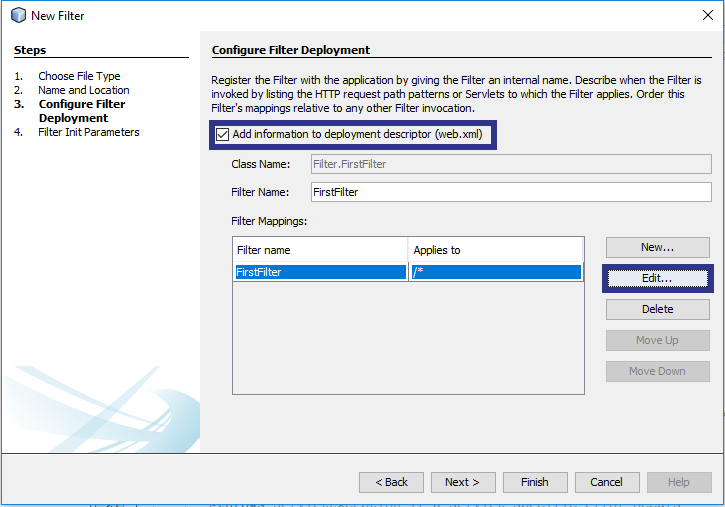
**Step: 8**

****

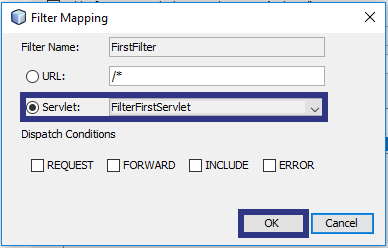
**Step: 9**

****

**Step: 9**

****

**Step: 10**

****

As we have mapped FirstFilter to FirstServlet as soon as we request for FirstServlet, FirstFilter will be executed.

After Ok click **Finish**

**Step: 11**

In FirstFilter only following code is needed.

package Filter;

import java.io.IOException;

import javax.servlet.Filter;

import javax.servlet.FilterChain;

import javax.servlet.FilterConfig;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

public class FirstFilter implements Filter {

public FirstFilter() {

}

private void doBeforeProcessing(ServletRequest request, ServletResponse response)

throws IOException, ServletException {

System.out.println("FirstFilter-In request Filter" );

}

private void doAfterProcessing(ServletRequest request, ServletResponse response)

throws IOException, ServletException {

System.out.println("FirstFilter-In response Filter" );

}

public void doFilter(ServletRequest request, ServletResponse response,

FilterChain chain)

throws IOException, ServletException {

doBeforeProcessing(request, response);

chain.doFilter(request, response);

doAfterProcessing(request, response);

}

public void destroy() {

}

public void init(FilterConfig filterConfig) {

}

}

**Explanation of code**

In filter doFilter is most important method, in which we have chain.doFilter(request, response) function which is used to separate request filter and response filter. Anything written above chain.doFilter(request, response) is consider as request filter and anything written below chain.doFilter(request, response) is consider as response filter. For simplicity we can have code as follow.

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

throws IOException, ServletException

{

System.out.println("FirstFilter-In request Filter" );

chain.doFilter(request, response);

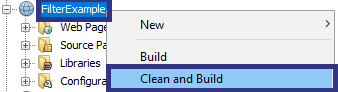
System.out.println("FirstFilter-In response Filter" );

}

Here we have simply printed message in server. Now, if we have more code of request and response filter than writing code in this manner make it complex, as solution we have two function doBeforeProcessing(request, response) and doAfterProcessing(request, response) which are used for request and response filter.

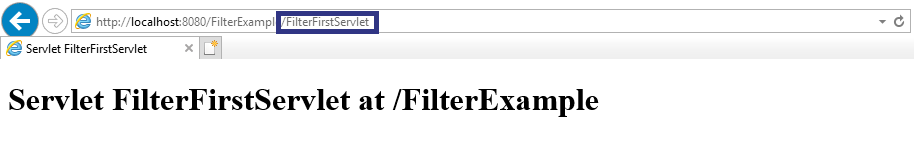
**Step: 12**

Build Application > Run Application

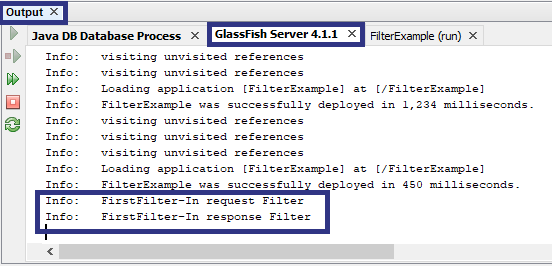


**Step: 13**

Request FilterFirstServlet



**Output**

****

In this example we have simply printed message in FirstFilter.

**Filter Chaining**

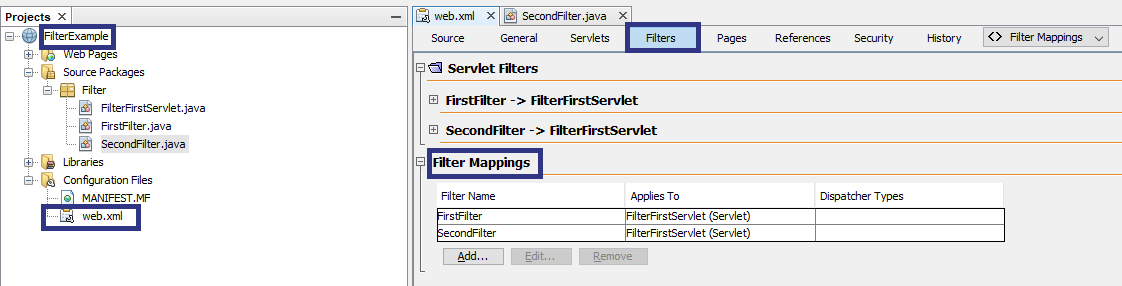
Filter chaining is sequence of filter which we can call for one or more servlet. Filter mapping allow to map more than one filter to particular servlet.

**Example:**

In FilterExample project create another filter named SecondFilter.

In SecondFilter print same message as we did in FirstServlet.

You can change mapping of filter from web.xml



So now if we will request FilterFirstServlet than output will be as below.

**Output**

