Lab 04

Introduction to Filters and Annotations

Mục tiêu

- Thực hành với Filters
- Một số annotation hay sử dụng, thay thế configurations trong các file config bằng annotations

Phần I Bài tập step by step

Bài 1.2

- Thực hành tạo Authentication Filter

Context để xây dựng Authentication Filter: khi user muốn truy cập vào một số nội dung hạn chế, ví dụ download tài liệu, administration, ... Trước khi truy cập tới resources (jsp, servlet,...) authentication filter sẽ kiểm tra trạng thái login, yêu cầu người dùng đăng nhập rồi mới chuyển tới resources yêu cầu.

STEP 1: Tạo mới project

Open Netbeans -> New project -> Web Application

Project name: AuthFilter

Server: Glassfish 4

Finish

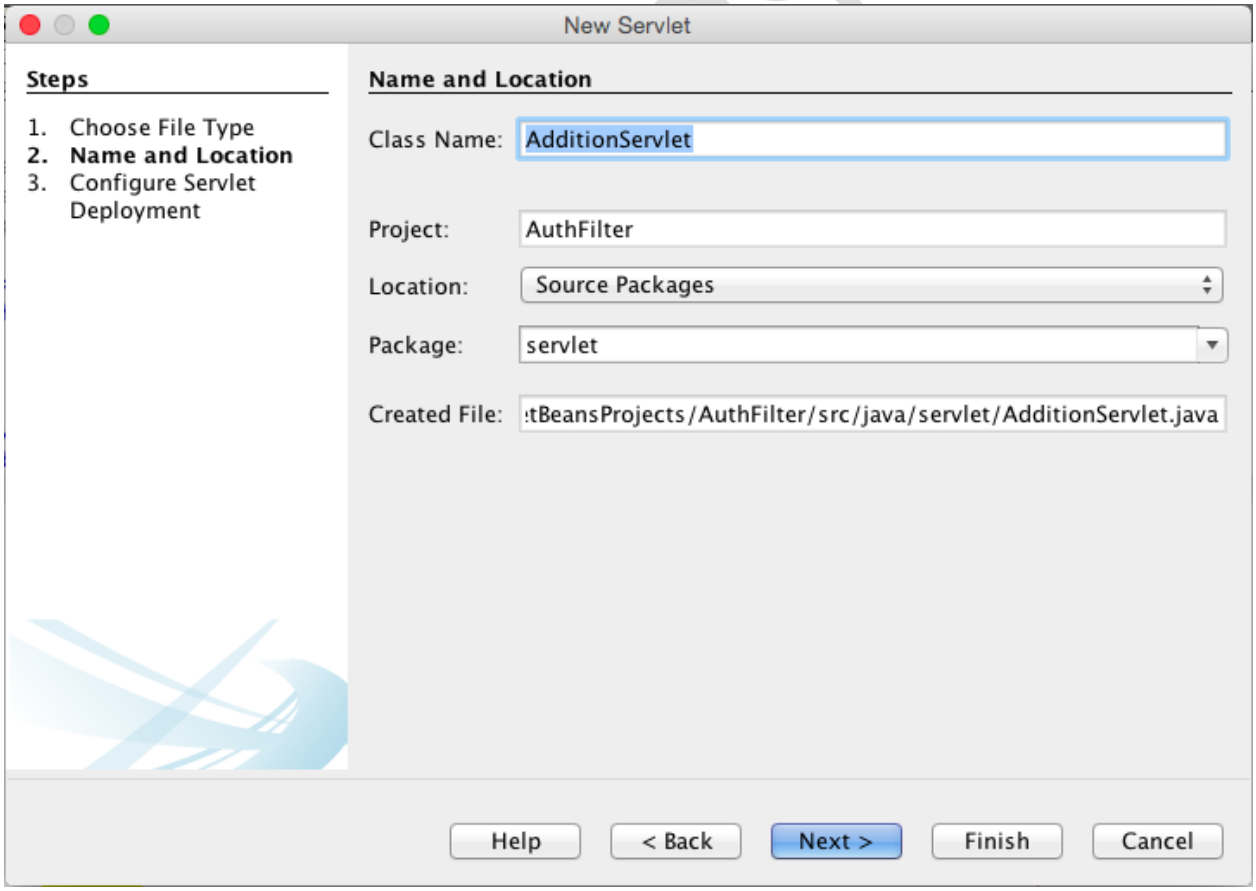
STEP 2: Tạo trang view

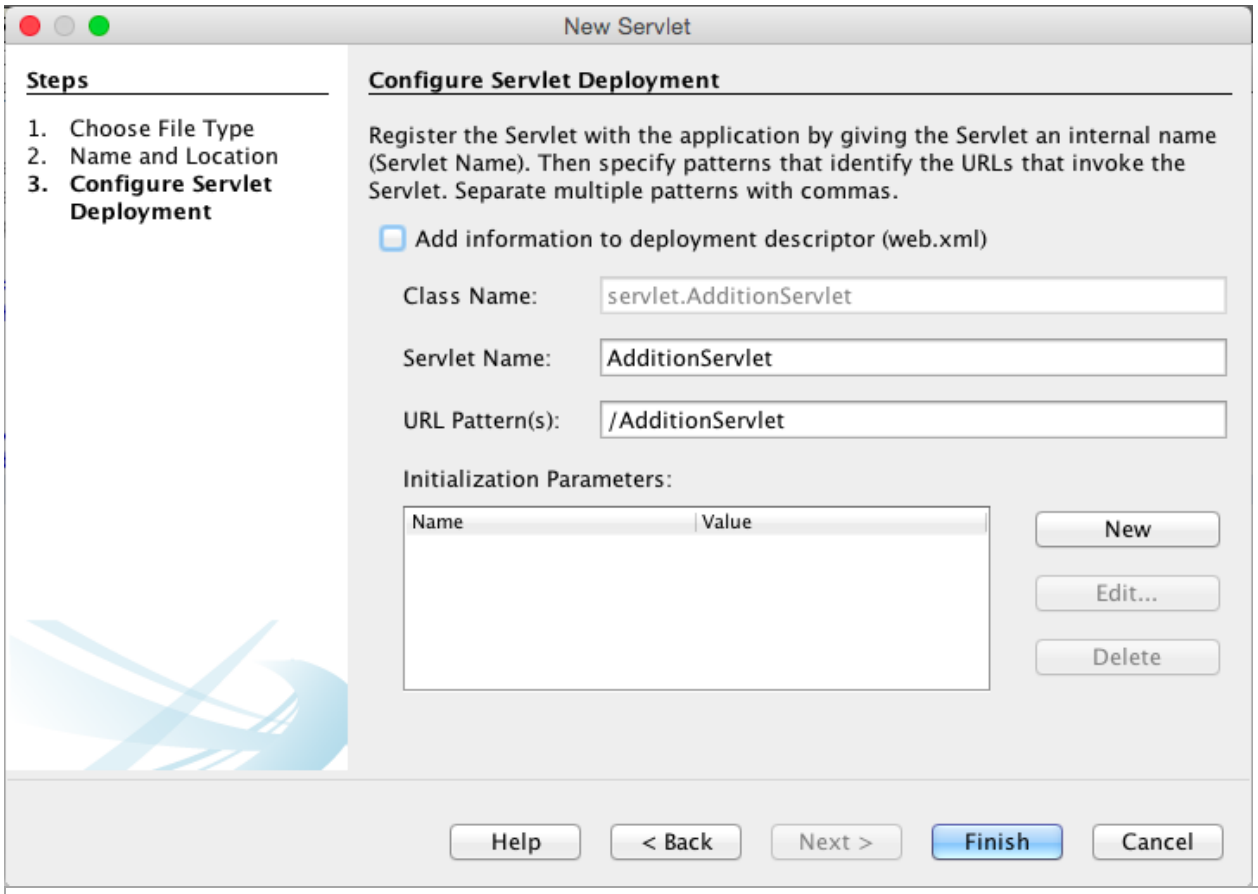
File index.jsp chứa form đơn giản cho phép nhập hai dữ liệu để tính tổng.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <title>Do some maths</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
   </head>
   <body>
      <h1>Entering values for doing something maths</h1>
      <form method="post" action="AdditionServlet">
```

```
        <table>
          <tr>
            <td>Type your name:</td>
            <td><input type="text" id="name" name="name"/></td>
          </tr>
          <tr>
            <td>First Value:</td>
            <td><input type="text" id="value1" name="value1"/></td>
          </tr>
          <tr>
            <td>Second value:</td>
            <td><input type="text" id="value2" name="value2" /></td>
          </tr>
          <tr>
            <td><input type="submit" value="Total" /></td>
          </tr>
        </table>
      </form>
    </body>
  </html>
```

STEP 3: Tạo **AdditionServlet** thực hiện cộng tổng hai giá trị lấy về từ form






```
package servlet;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author HAITHANH
 */
@WebServlet(name = "AdditionServlet", urlPatterns = {"/AdditionServlet"})
public class AdditionServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
```

```
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        int value1;
        int value2;
        int total = 0;
        String name = request.getParameter("name");
        String warning = "";

        try {
            value1 = Integer.parseInt(request.getParameter("value1"));
            value2 = Integer.parseInt(request.getParameter("value2"));
        } catch (NumberFormatException e) {
            value1 = 0;
            value2 = 0;
            warning = "We got some bad value(blank or non numerics values, we set 0 instead";
        }

        request.setAttribute("name", name);
        request.setAttribute("warning", warning);

        total = value1 + value2;

        request.setAttribute("total", total);

        request.getRequestDispatcher("show.jsp").forward(request, response);
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the
left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
```

```
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

STEP 4: Tạo trang show.jsp để hiển thị kết quả trả về của AdditionServlet

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Result</title>
    </head>
    <body>
        Thanks ${name} for passing by. <br/>
        The total is: ${total}. <br/>
        <br/>${warning}
    </body>
</html>
```

Chạy thử ứng dụng để test.

STEP 5: Tạo authentication filter để check trước khi forward tới resources

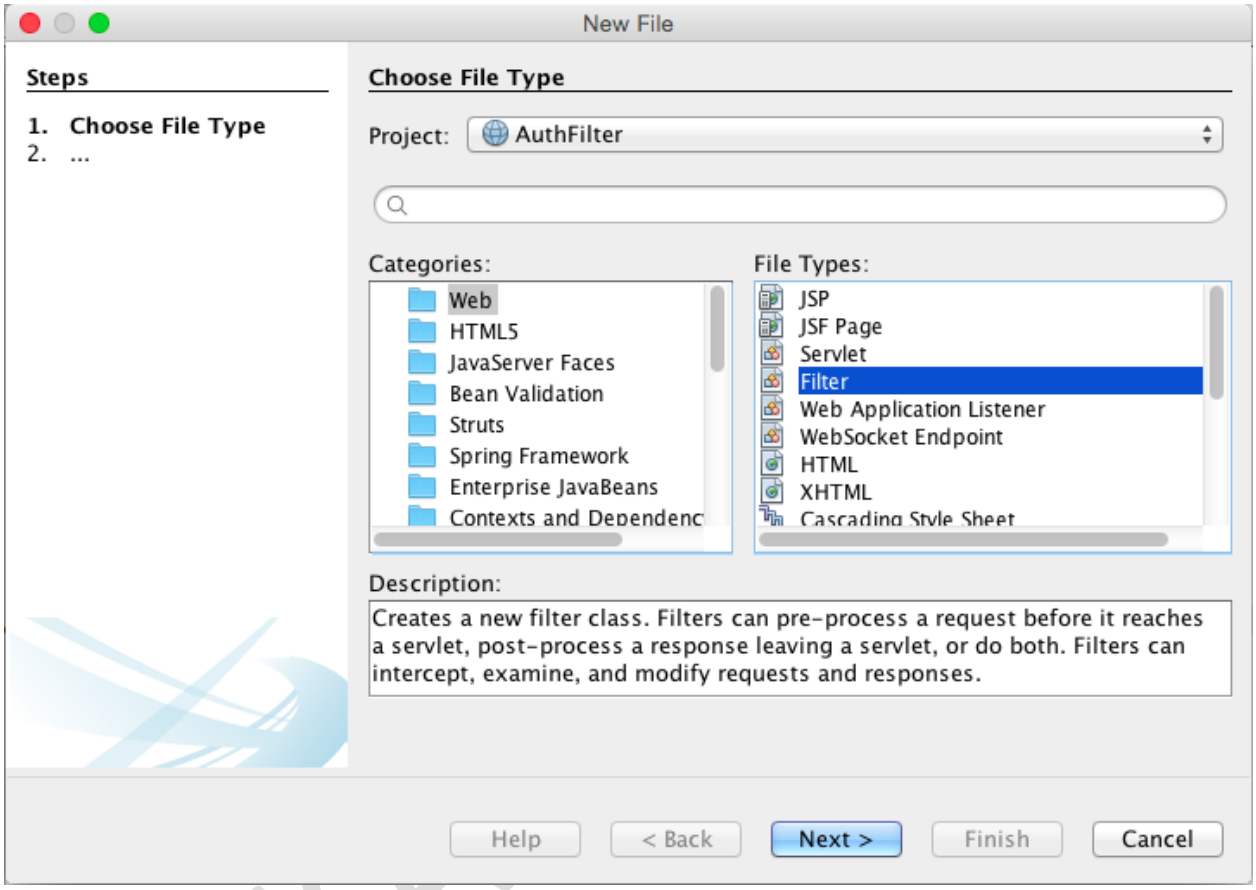
- Click chuột phải vào project chọn "New / Other / Web / Filter"
- Đặt tên file: **AuthenticationFilter**, package **filter**

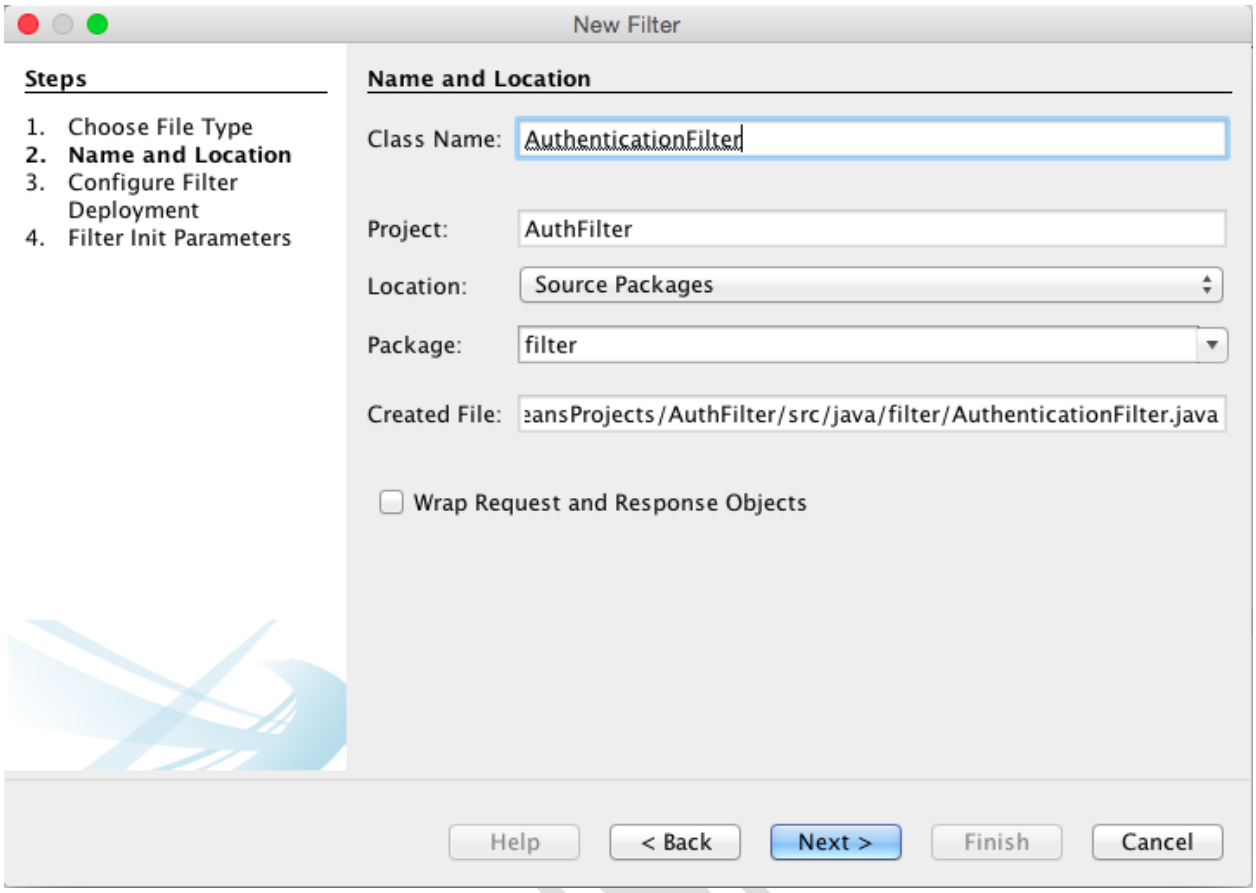
Trong vùng **Filter Mappings:**

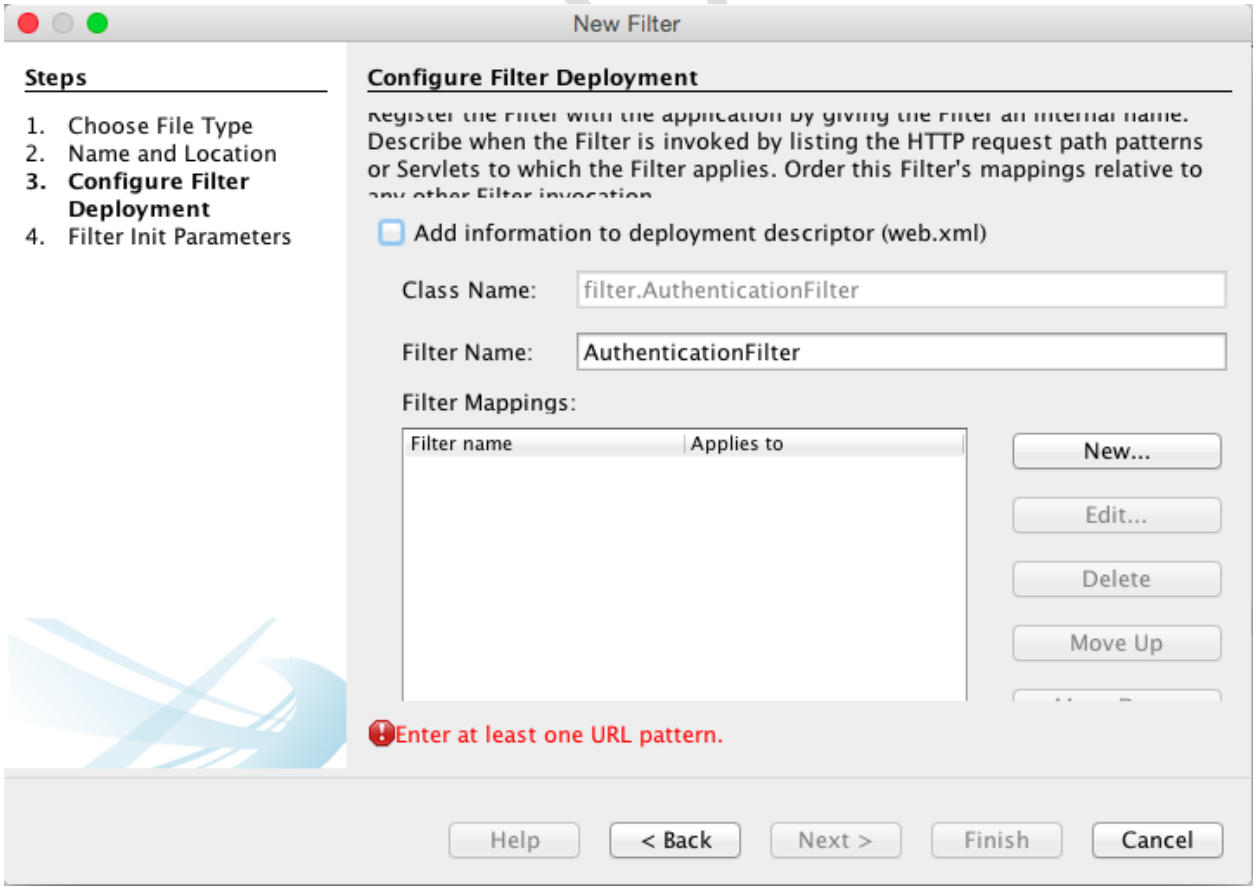
- Chọn: New /AdditionServlet
- Dispatch Conditions: REQUEST, FORWARD
- Thêm initial parameter + Next

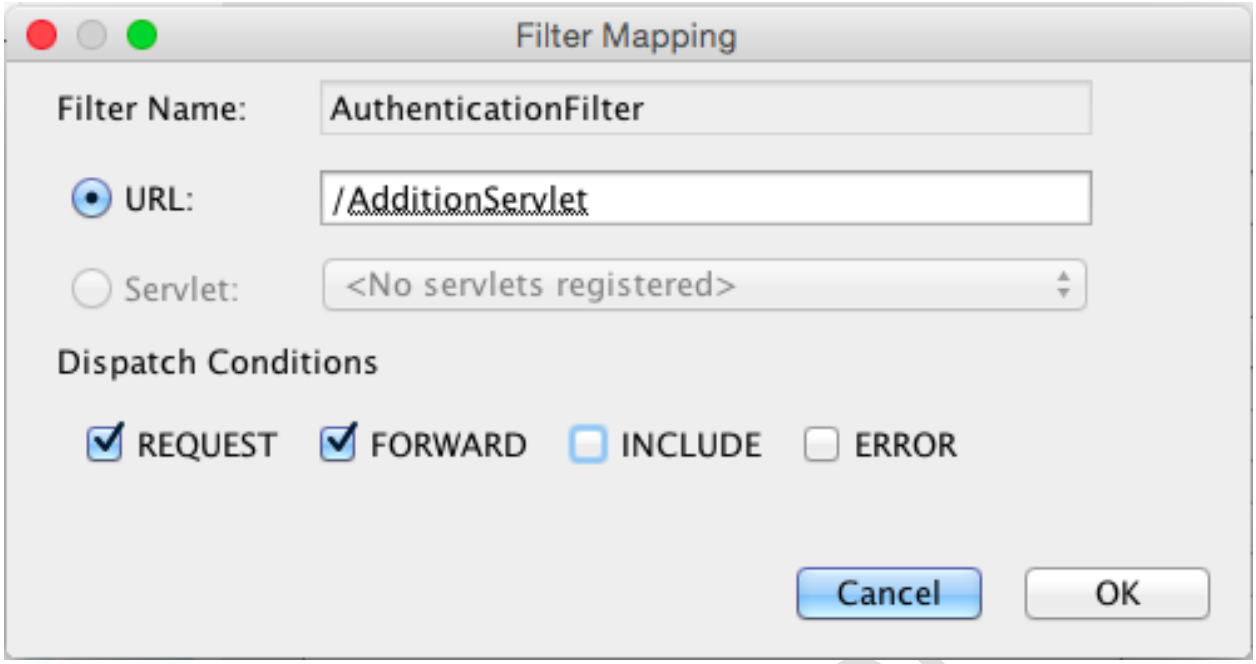
Trong vùng **Initialization Parameters:**

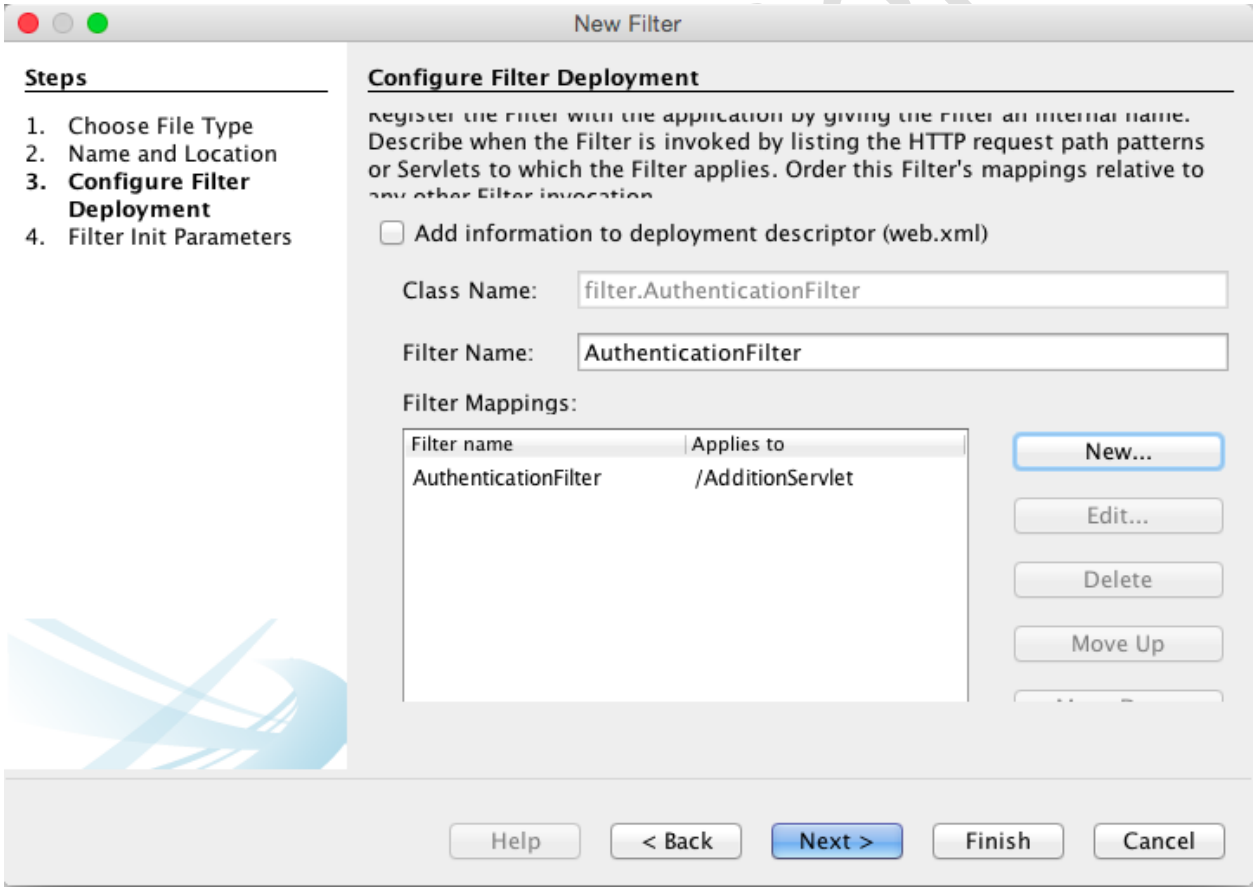
- Chọn: New
- Name: loginActionURI
- Value: /AdditionServlet/LoginServlet




New Filter
Steps
1. Choose File Type
2. Name and Location
3. Configure Filter Deployment
4. Filter Init Parameters
Name and Location
Class Name: AuthenticationFilter
Project: AuthFilter
Location: Source Packages
Package: filter
Created File: eansProjects/AuthFilter/src/java/filter/AuthenticationFilter.java
Wrap Request and Response Objects
Help
< Back
Next >
Finish
Cancel


New Filter
Steps
1. Choose File Type
2. Name and Location
3. Configure Filter Deployment
4. Filter Init Parameters
Configure Filter Deployment
Describe when the Filter is invoked by listing the HTTP request path patterns or Servlets to which the Filter applies. Order this Filter's mappings relative to
Add information to deployment descriptor (web.xml)
Class Name: filter.AuthenticationFilter
Filter Name: AuthenticationFilter
Filter Mappings:
Filter name
Applies to
New...
Edit...
Delete
Move Up
Enter at least one URL pattern.
Help
< Back
Next >
Finish
Cancel





STEP 6: Tạo bean User

- Click chuột phải vào project chọn "New /Java Class..". Đặt tên lớp User, package model

```
package model;

public class User {
    private String name = "UNKNOW";
```

```
    private String password = "UNKNOW";

    public User(String name, String password){
        this.name = name;
        this.password = password;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;

   }

}
```

- Thêm code sau vào filter: "**AuthenticationFilter**":

```
package filter;

import java.io.IOException;
import javax.servlet.DispatcherType;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.annotation.WebInitParam;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import model.User;


@WebFilter(filterName = "AuthenticationFilter", urlPatterns =
{"/AdditionServlet"}, dispatcherTypes = {DispatcherType.REQUEST,
DispatcherType.FORWARD}, initParams = {
    @WebInitParam(name = "loginActionURI", value =
"/AdditionServlet/LoginServlet")})
public class AuthenticationFilter implements Filter {

    // The filter configuration object we are associated with.  If
    // this value is null, this filter instance is not currently
    // configured.
```

```
    private FilterConfig filterConfig = null;

    private String LOGIN_ACTION_URI;

    public AuthenticationFilter() {
    }

    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain)
            throws IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest) request;
        HttpSession session = req.getSession();
        User user = (User) session.getAttribute("user");

        if (user == null && !LOGIN_ACTION_URI.equals(req.getRequestURI())){
            RequestDispatcher rd = req.getRequestDispatcher("/login.jsp");
            rd.forward(request, response);
            return;
        }

        chain.doFilter(request, response);
    }

    public void destroy() {
    }


    public void init(FilterConfig filterConfig) {
        this.filterConfig = filterConfig;
        LOGIN_ACTION_URI = this.filterConfig.getInitParameter("loginActionURI");
    }
}
```

STEP 7: Tạo servlet LoginServlet và các trang jsp phục vụ login

Trong ví dụ này chúng ta tạo một bảng map <name, user>. Có thể thay thế bảng map tự tạo này bằng cách get all users từ database.

Tạo **LoginServlet:**

```
package servlet;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import model.User;
```

```java
@WebServlet(name = "LoginServlet", urlPatterns = {"/LoginServlet"})
public class LoginServlet extends HttpServlet {

    private static final Map<String, User> users = getUsers();

    private static final Map<String, User> getUsers() {
        //Co the thay bang get all users from database
        User u1 = new User("peter", "12345");
        User u2 = new User("hana", "123$$");

        Map<String, User> users = new HashMap<String, User>();
        users.put(u1.getName(), u1);
        users.put(u2.getName(), u2);

        return users;
    }

    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        doPost(req, res);
    }

    public void doPost(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        RequestDispatcher rd;
        String name = req.getParameter("username");
        String password = req.getParameter("password");

        User user = validateLogin(name, password);

        if (user == null){
            rd = req.getRequestDispatcher("/loginError.jsp");
        }
        else{
            HttpSession session = req.getSession();
            session.setAttribute("user", user);
            rd = req.getRequestDispatcher("/index.jsp");
        }

        rd.forward(req, res);
    }

    private User validateLogin(String name, String password) {
        // All parameters must be valid
        if (name == null || password == null){
            return null;
        }

        // Get a user by key
        User user = users.get(name);

        if (user == null){
```

```
            return null;
        }

        // Check if the password is valid
        if (!user.getPassword().equals(password.trim())){
            return null;
        }

        return user;
    }

}
```

Tạo trang login.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Login</title>
  </head>
  <body>
    <form method="post" action="LoginServlet">
      <table>
        <tr>
          <td>Username:</td>
          <td><input type="text" id="username" name="username"/></td>
        </tr>
        <tr>
          <td>Password:</td>
          <td><input type="password" id="password" name="password" /></td>
        </tr>
        <tr>
          <td><input type="submit" value="Login" /></td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

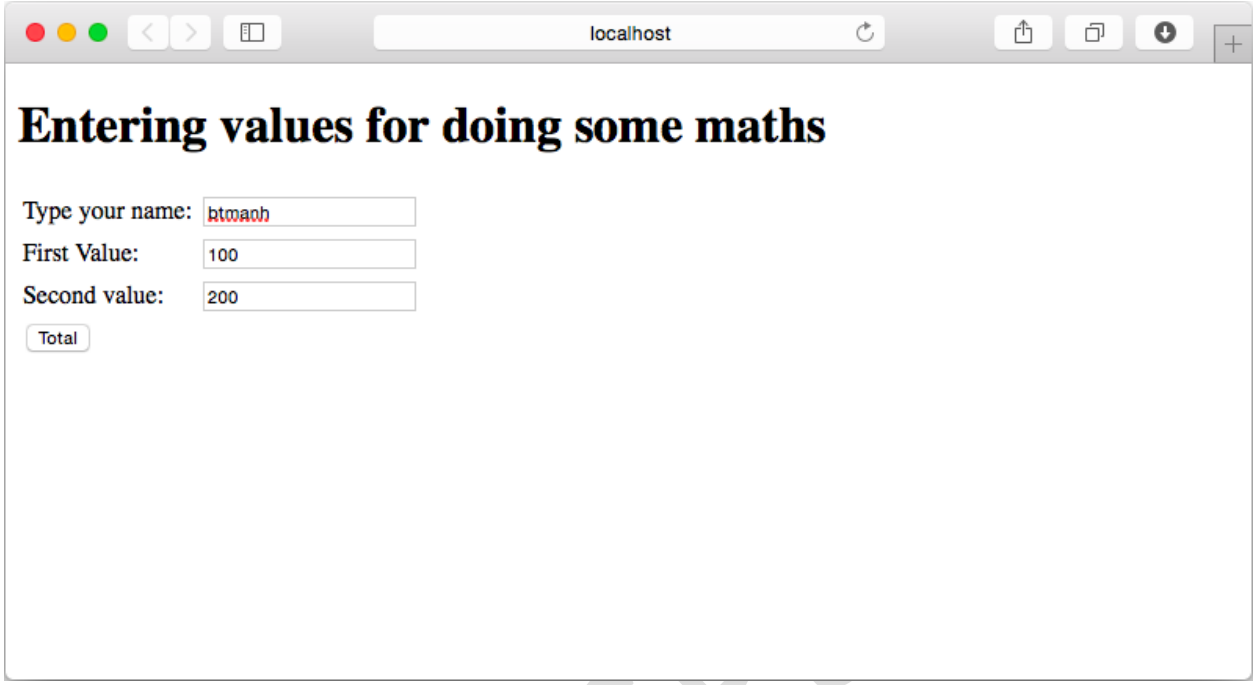
Trên trang login.jsp chứa form đăng nhập.

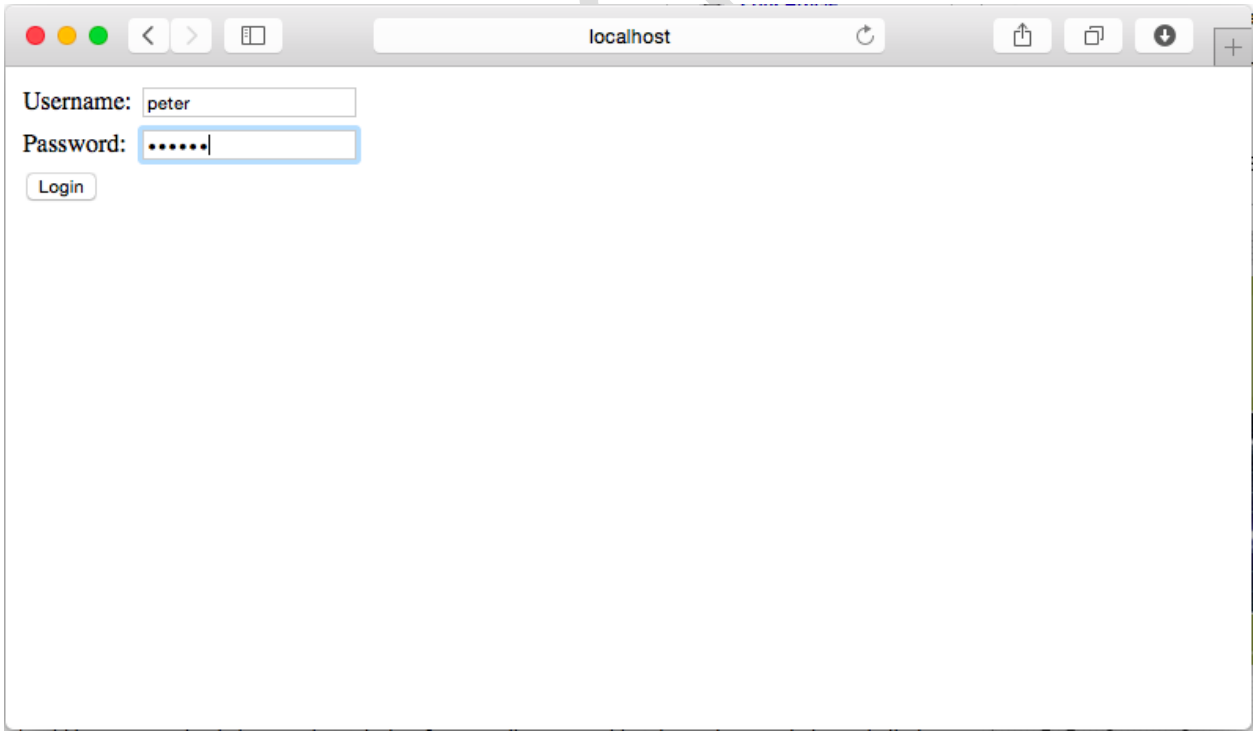
Tạo loginError.jsp:

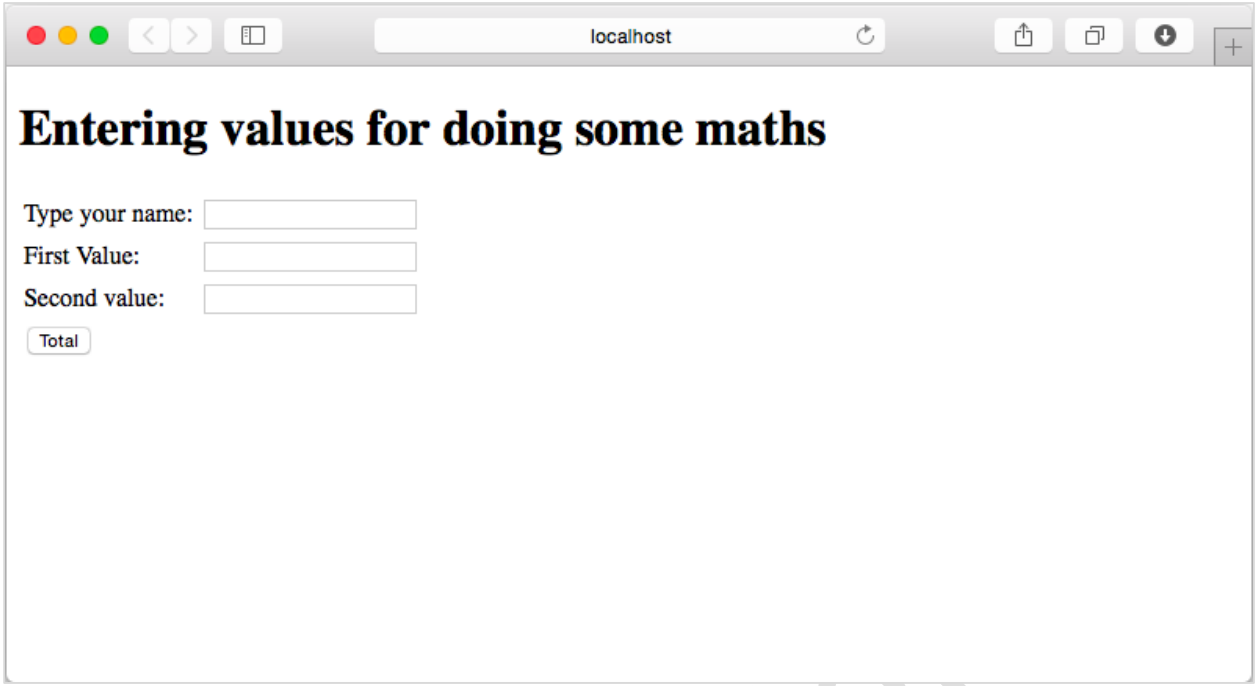
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title></title>
  </head>
  <body>
    <h1>Wrong username or password</h1>
  </body>
```

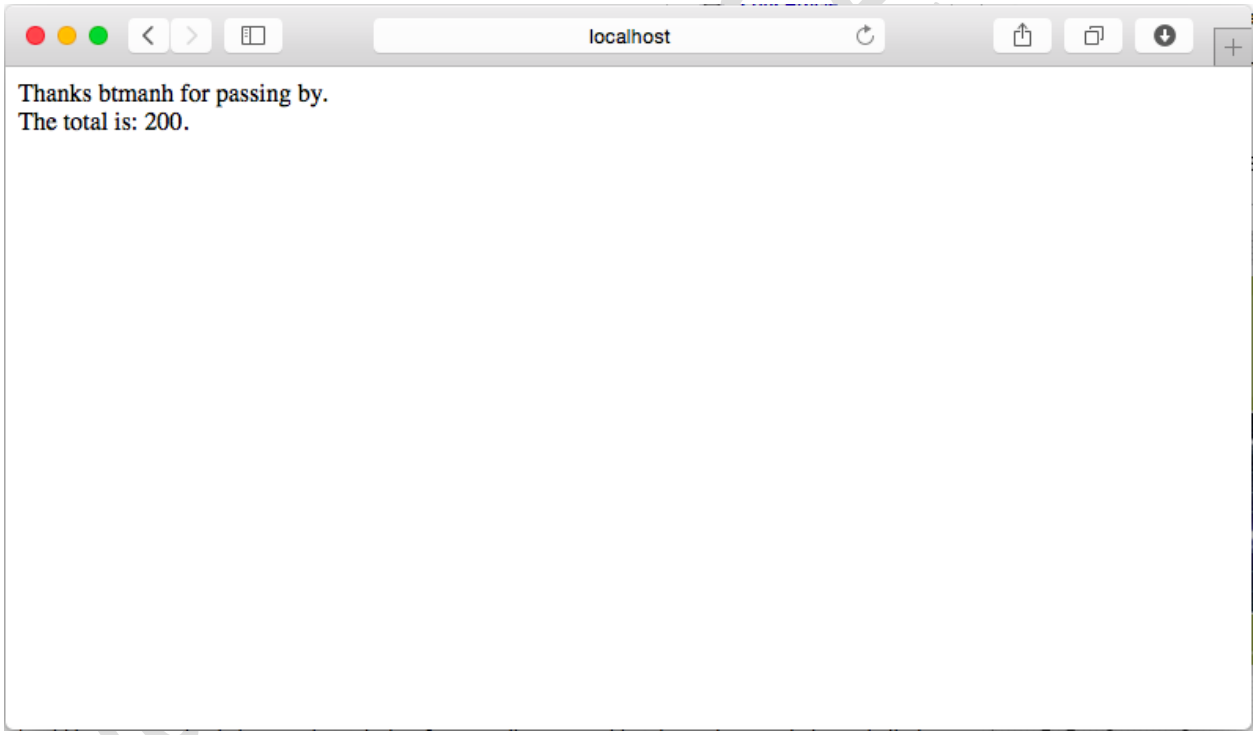
```
</html>
```

Chạy chương trình:









Bài 1.2

Mục tiêu:

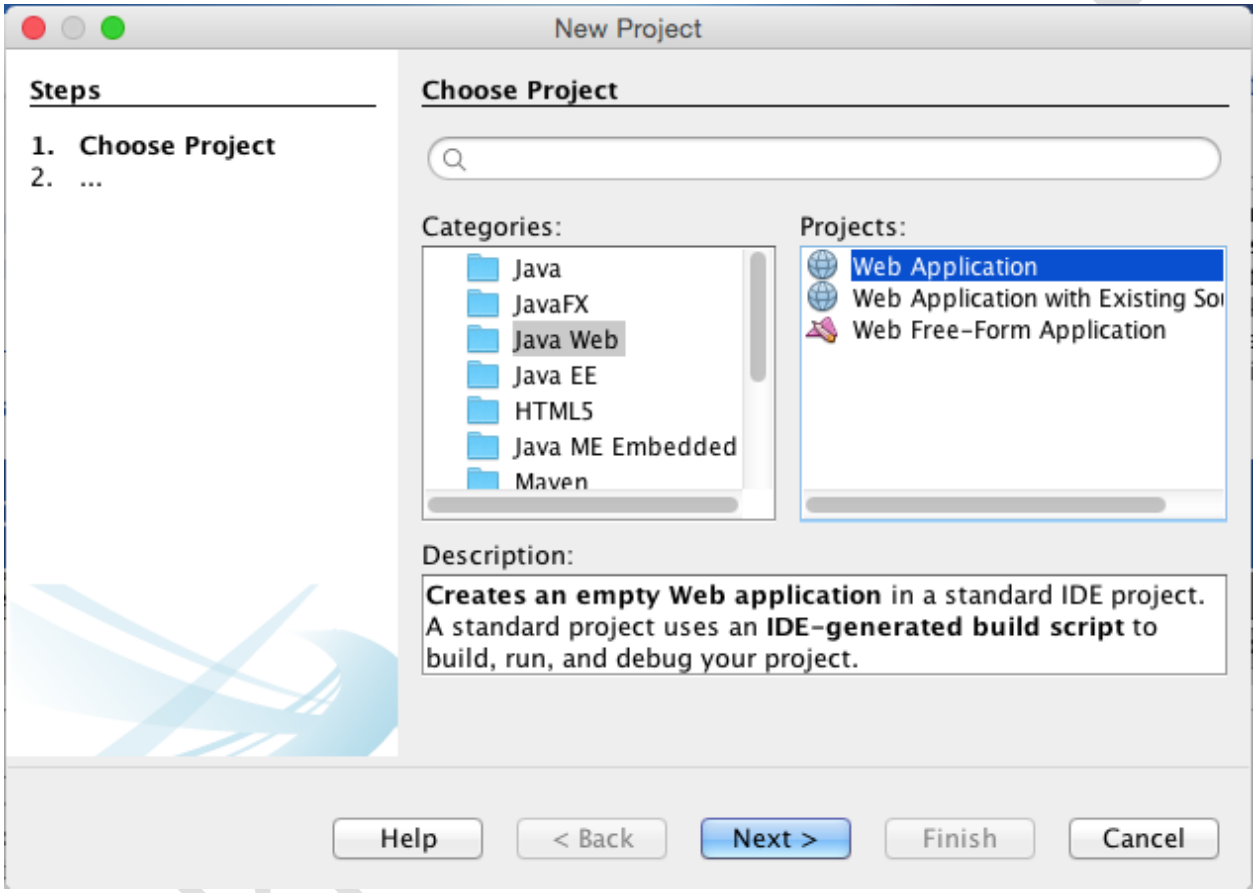
- Sử dụng Filter để thao tác trực tiếp với request và response
- Sử dụng HttpServletRequestWrapper và HttpServletResponseWrapper

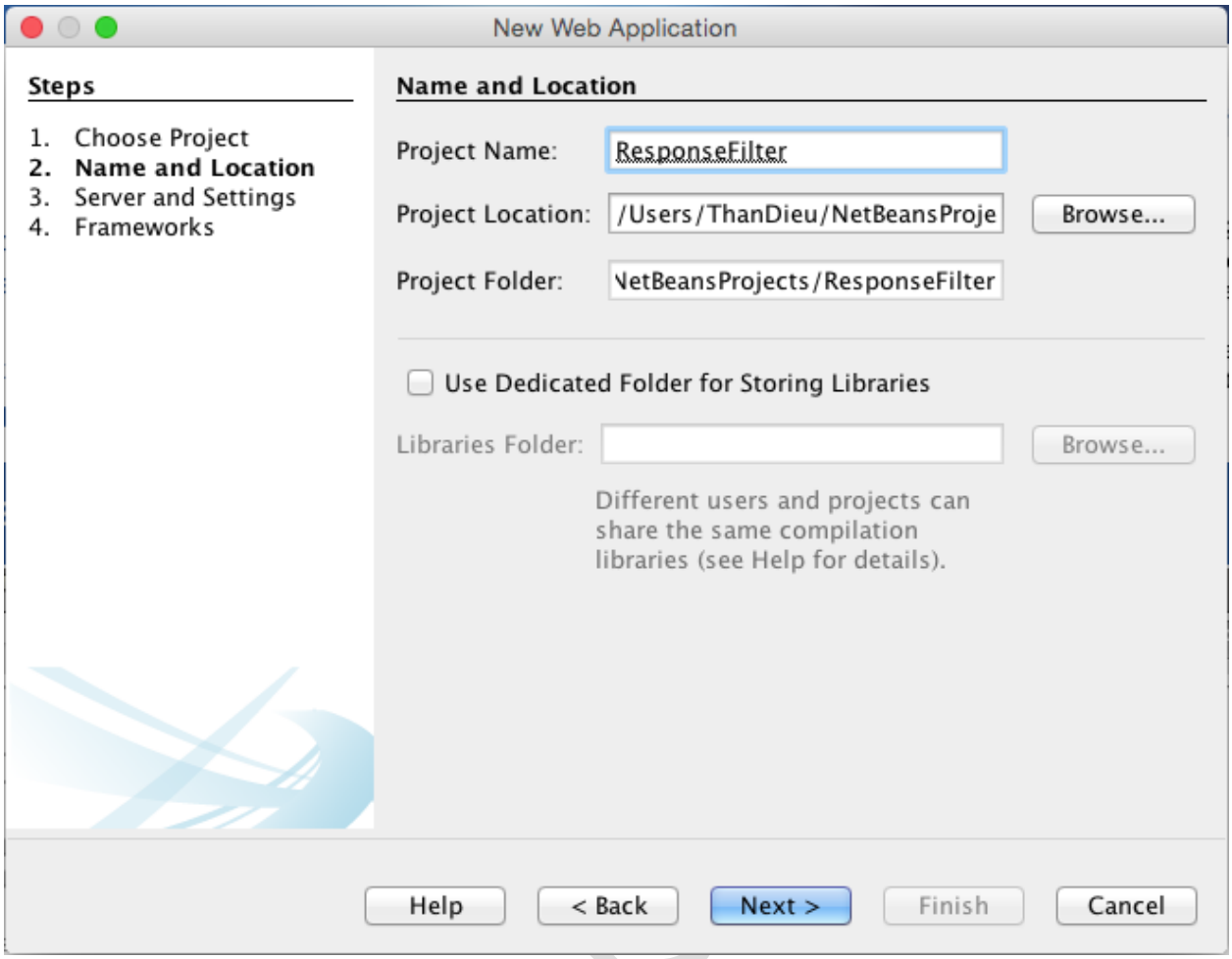
Thông thường Filter can thiệp vào request gửi từ client, xử lý request trước khi chuyển tiếp nó đến Servlet để được xử lý và tạo response trả về cho client. Tuy nhiên, ở mức độ (cao cấp) phức tạp hơn, filter có thể can thiệp vào response xây dựng bởi servlet và trả về cho client. Vấn đề phức tạp

ở đây là response tạo bởi Servlet sẽ được servlet chuyển thẳng đến cho client. Do đó để can thiệp vào response này, Filter cần phải tạo riêng response. Các interface wrapper của Servlet cho phép làm điều này.

Project sau sẽ thực hiện đếm số visitor 1 trang web thông qua filter (bằng cách đếm số lần servlet được gọi bởi client request). Filter sẽ can thiệp vào response servlet trả về bằng cách thêm vào response số lần mà nó đếm được.

STEP 1: Tạo web application ResponseFilter

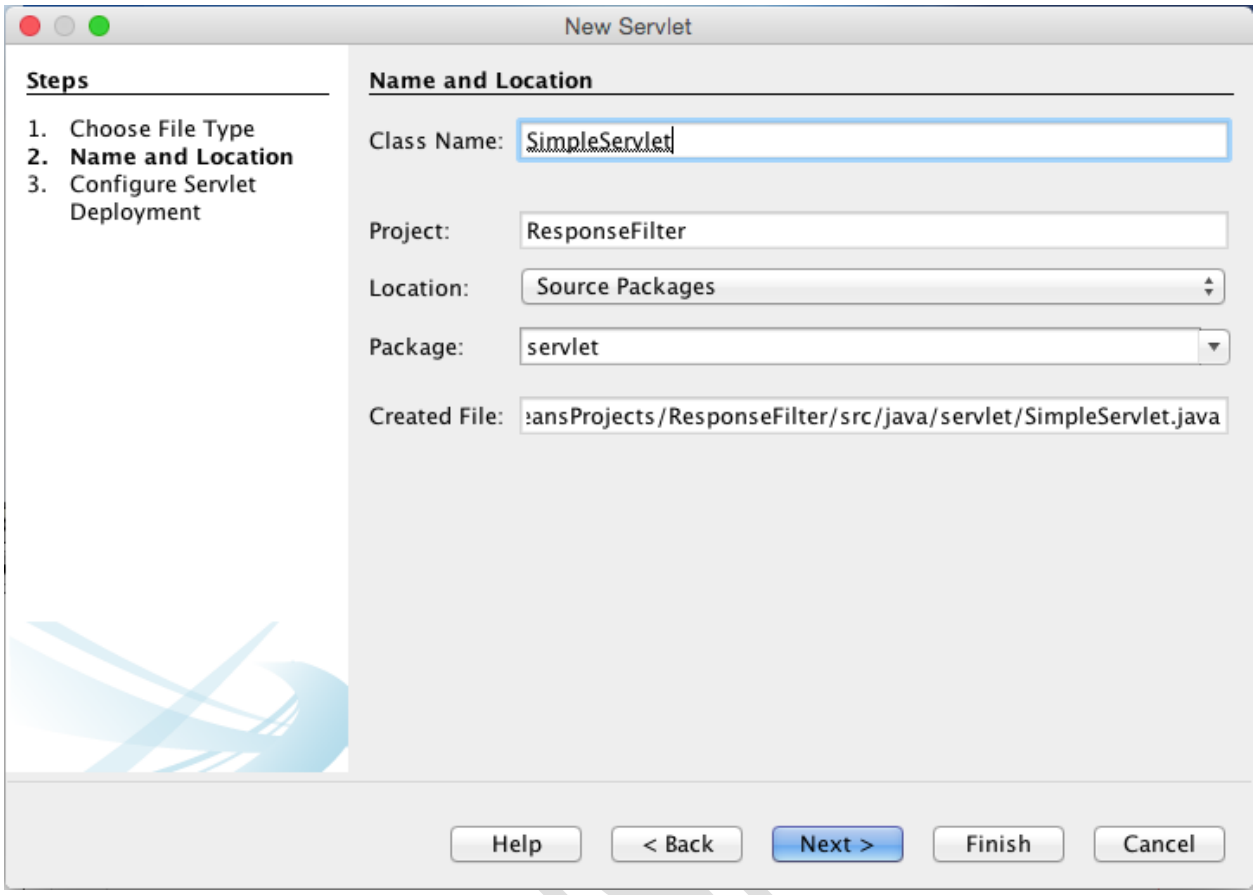


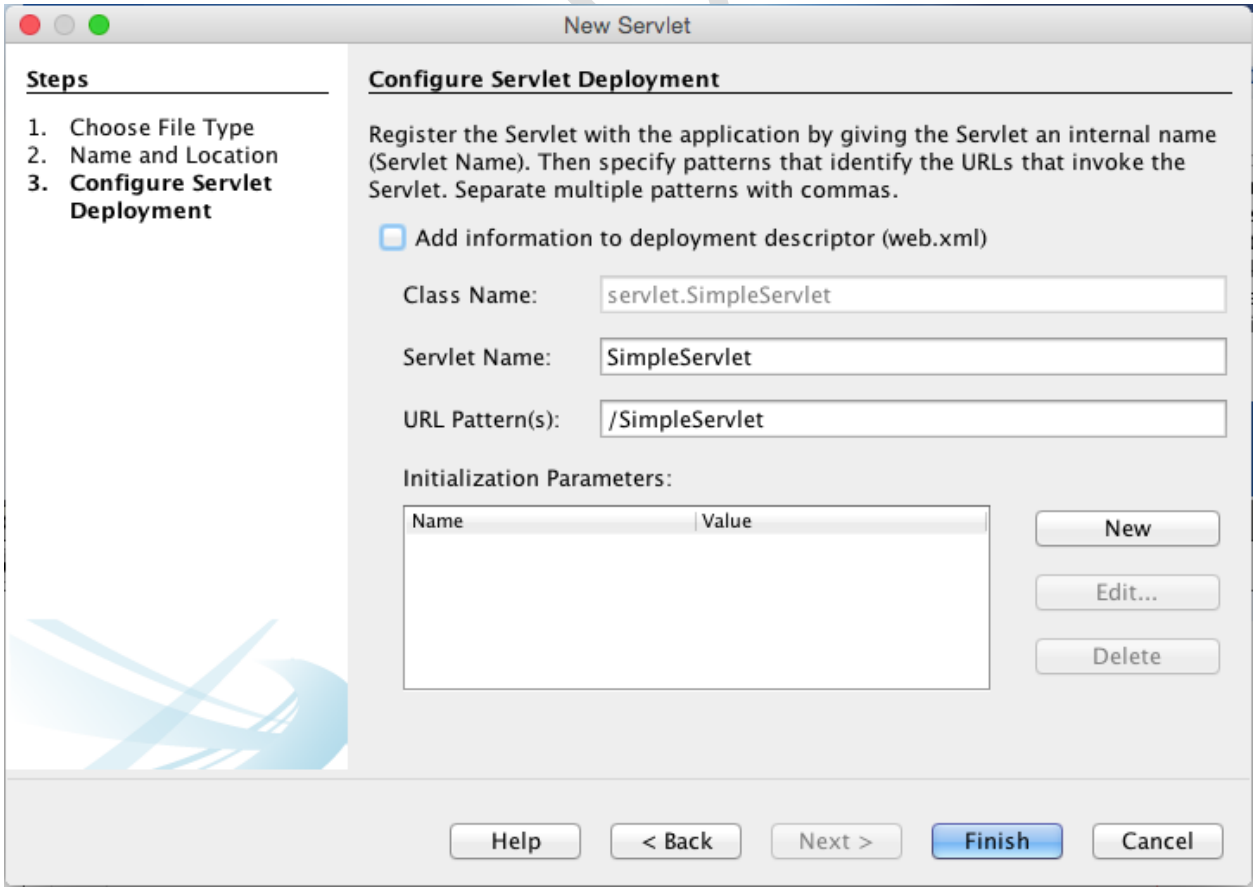


Chọn GlassFish server và Java EE 7 -> Finish.

STEP 2: Tạo SimpleServlet

Tạo package servlet -> tạo SimpleServlet.


New Servlet
Steps
1. Choose File Type
2. Name and Location
3. Configure Servlet Deployment
Name and Location
Class Name: SimpleServlet
Project: ResponseFilter
Location: Source Packages
Package: servlet
Created File: :ansProjects/ResponseFilter/src/java/servlet/SimpleServlet.java
Help
< Back
Next >
Finish
Cancel

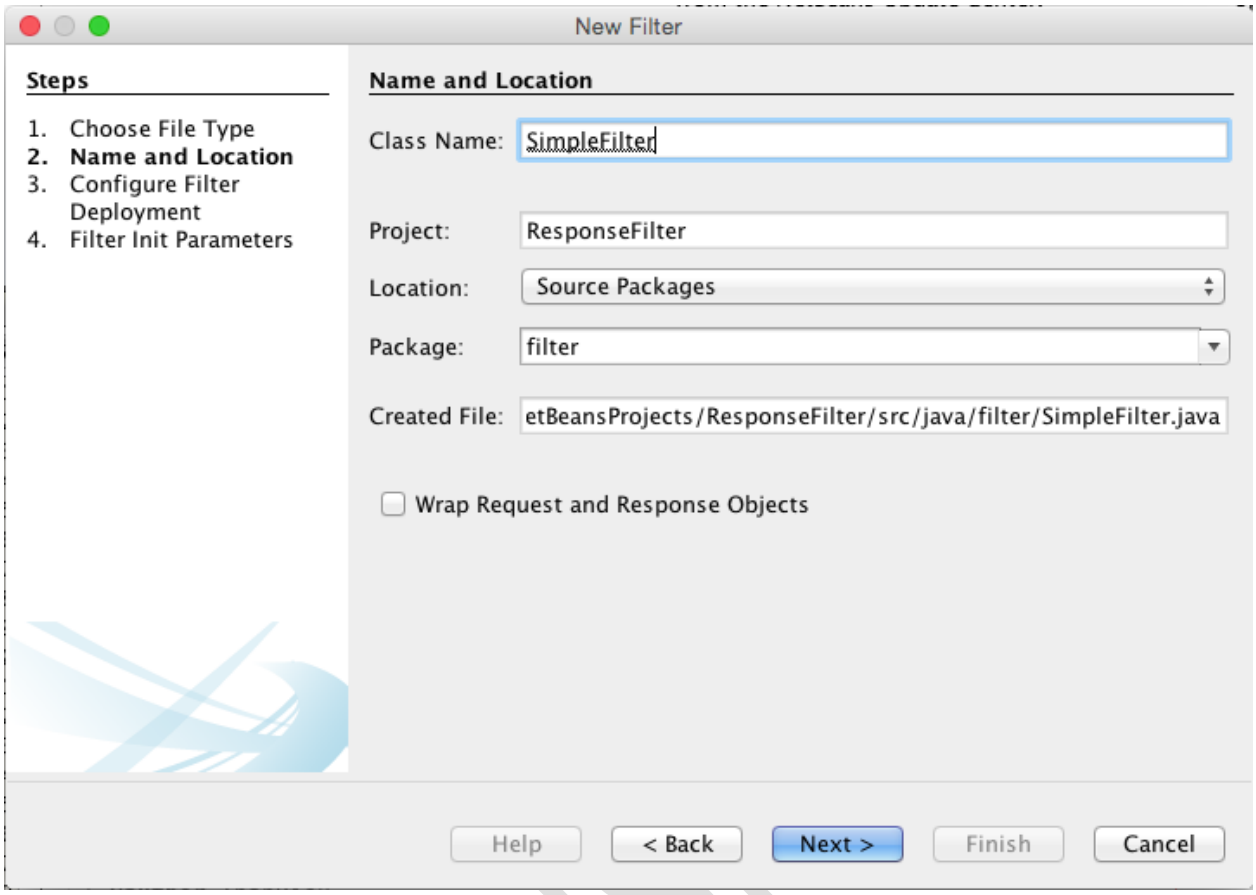

New Servlet
Steps
1. Choose File Type
2. Name and Location
3. Configure Servlet Deployment
Configure Servlet Deployment
Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.
Add information to deployment descriptor (web.xml)
Class Name: servlet.SimpleServlet
Servlet Name: SimpleServlet
URL Pattern(s): /SimpleServlet
Initialization Parameters:
Name
Value
New
Edit...
Delete
Help
< Back
Next >
Finish
Cancel

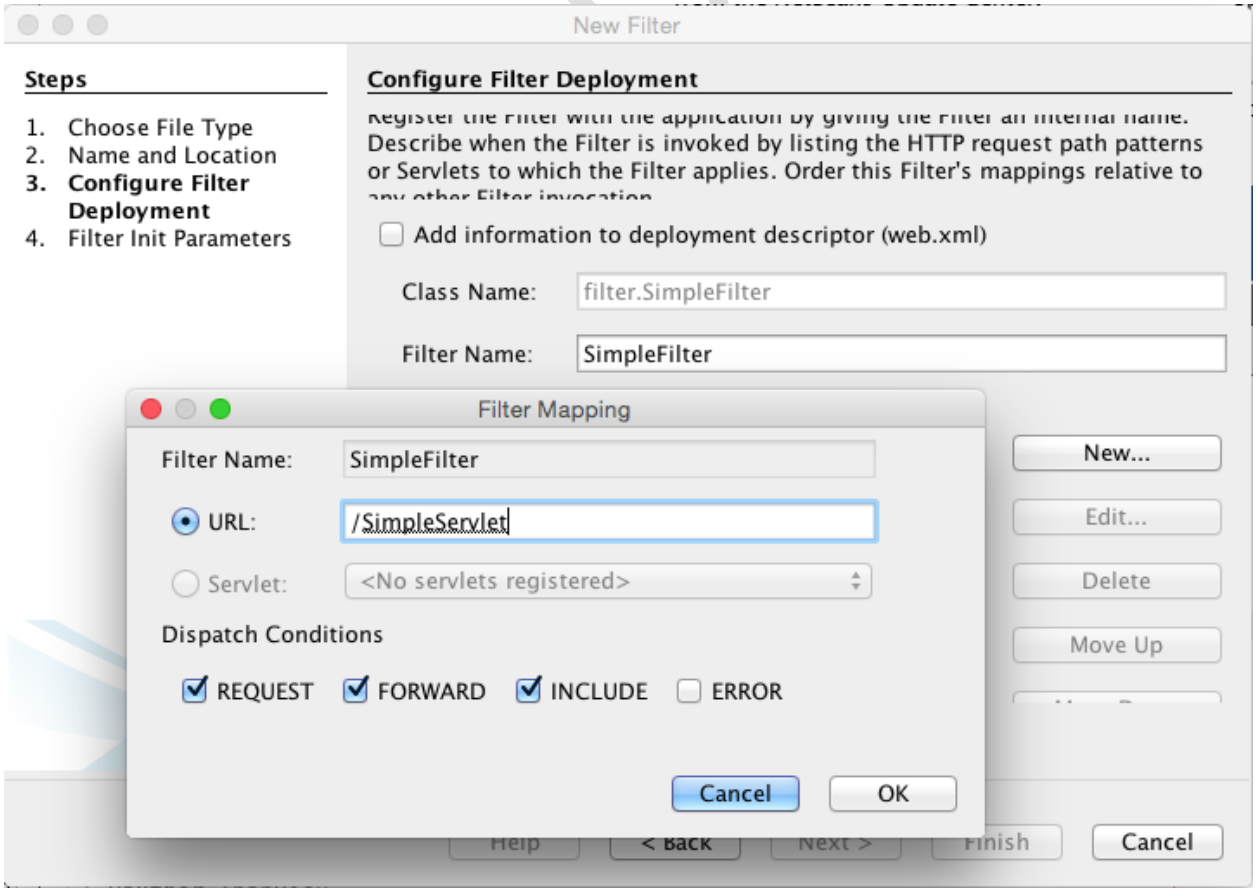
SimpleServlet.java:

```
package servlet;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author ThanDieu
 */
@WebServlet(name = "SimpleServlet", urlPatterns = {"/SimpleServlet"})
public class SimpleServlet extends HttpServlet {

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

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code.
*/
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet TestServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet TestServlet at " +
request.getContextPath() + "</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }

}
```

STEP 3: Tạo SimpleFilter thực hiện filter request.

Tạo package filter -> Tạo SimpleFilter


New Filter
Steps
1. Choose File Type
2. Name and Location
3. Configure Filter Deployment
4. Filter Init Parameters
Name and Location
Class Name: SimpleFilter
Project: ResponseFilter
Location: Source Packages
Package: filter
Created File: etBeansProjects/ResponseFilter/src/java/filter/SimpleFilter.java
Wrap Request and Response Objects
Help
< Back
Next >
Finish
Cancel


New Filter
Steps
1. Choose File Type
2. Name and Location
3. Configure Filter Deployment
4. Filter Init Parameters
Configure Filter Deployment
Describe when the Filter is invoked by listing the HTTP request path patterns or Servlets to which the Filter applies. Order this Filter's mappings relative to
Add information to deployment descriptor (web.xml)
Class Name: filter.SimpleFilter
Filter Name: SimpleFilter
New...
Edit...
Delete
Move Up
Filter Mapping
Filter Name: SimpleFilter
URL: /SimpleServlet
Servlet: <No servlets registered>
Dispatch Conditions
REQUEST
FORWARD
INCLUDE
ERROR
Cancel
OK
Cancel

OK -> Finish

```
package filter;

import java.io.IOException;
import java.util.Date;
import javax.servlet.DispatcherType;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;


@WebFilter(filterName = "SimpleFilter", urlPatterns = {"/SimpleServlet"},
         dispatcherTypes = {DispatcherType.REQUEST,
                            DispatcherType.FORWARD,
                            DispatcherType.INCLUDE})
public class SimpleFilter implements Filter {

    private FilterConfig filterConfig = null;

    private ServletContext context;

    public SimpleFilter() {
    }

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        this.filterConfig = filterConfig;
        context = filterConfig.getServletContext();
    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws IOException, ServletException {
        System.out.println("Within the SimpleFilter...");
        System.out.println("Filtering the request of client...");

        HttpServletRequest req = (HttpServletRequest) request;
        String requestURL = req.getRequestURL().toString();
        context.log("Requesting URL: " + requestURL + "Time: " + new Date());

        chain.doFilter(request, response);

        System.out.println("Within the SimpleFilter...");
        System.out.println("Filtering the response...");
    }

    @Override
    public void destroy() {
```

```
    }

}
```

STEP 4: Tao WrapperResponse

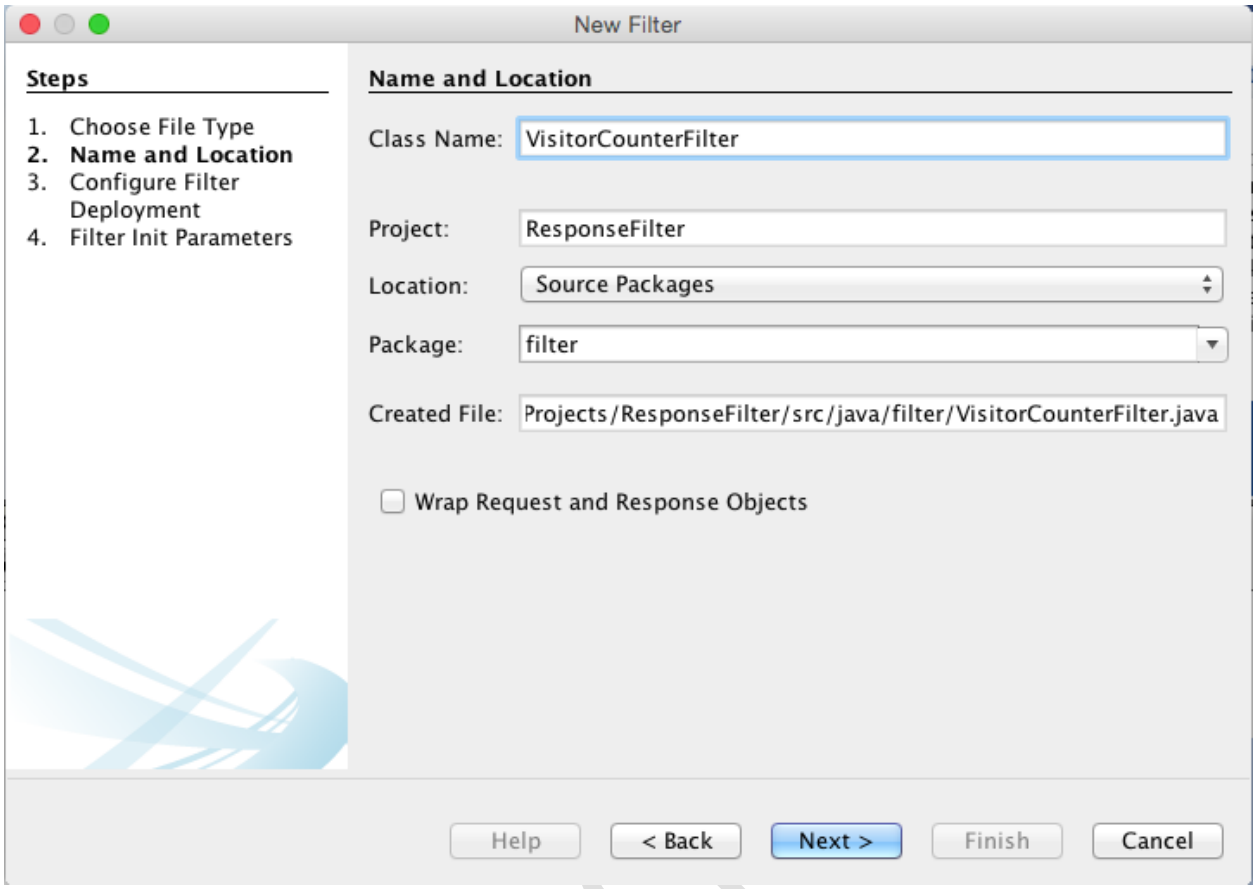
Filter muốn can thiệp vào response tạo ra bởi servlet trước khi trả về cho client thì chúng ta cần tạo một response mới cho Filter, extends HttpServletResponseWrapper.

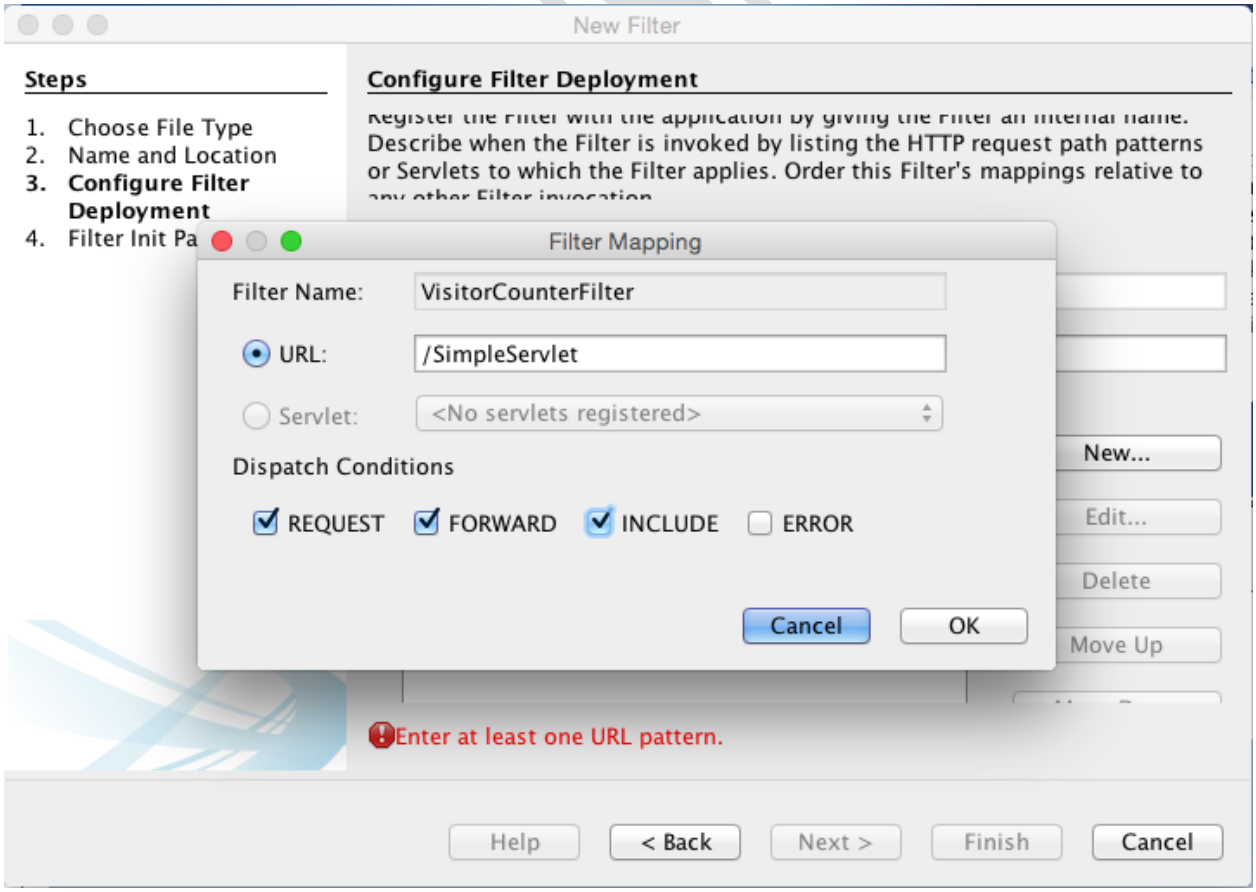
Trong response này, cần có một buffer để write out ra PrintWriter.

Tạo package util -> tạo FilterResponseWrapper

```
package util;

import java.io.CharArrayWriter;
import java.io.PrintWriter;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpServletResponseWrapper;

public class FilterResponseWrapper extends HttpServletResponseWrapper{

    private CharArrayWriter buffer;

    public FilterResponseWrapper(HttpServletResponse response) {
        super(response);
        buffer = new CharArrayWriter();
    }

    @Override
    public String toString(){
        return buffer.toString();
    }

    @Override
    public PrintWriter getWriter(){
        return new PrintWriter(buffer);
    }

}
```

STEP 5: Tao VisitorCounterFilter






```
package filter;
```

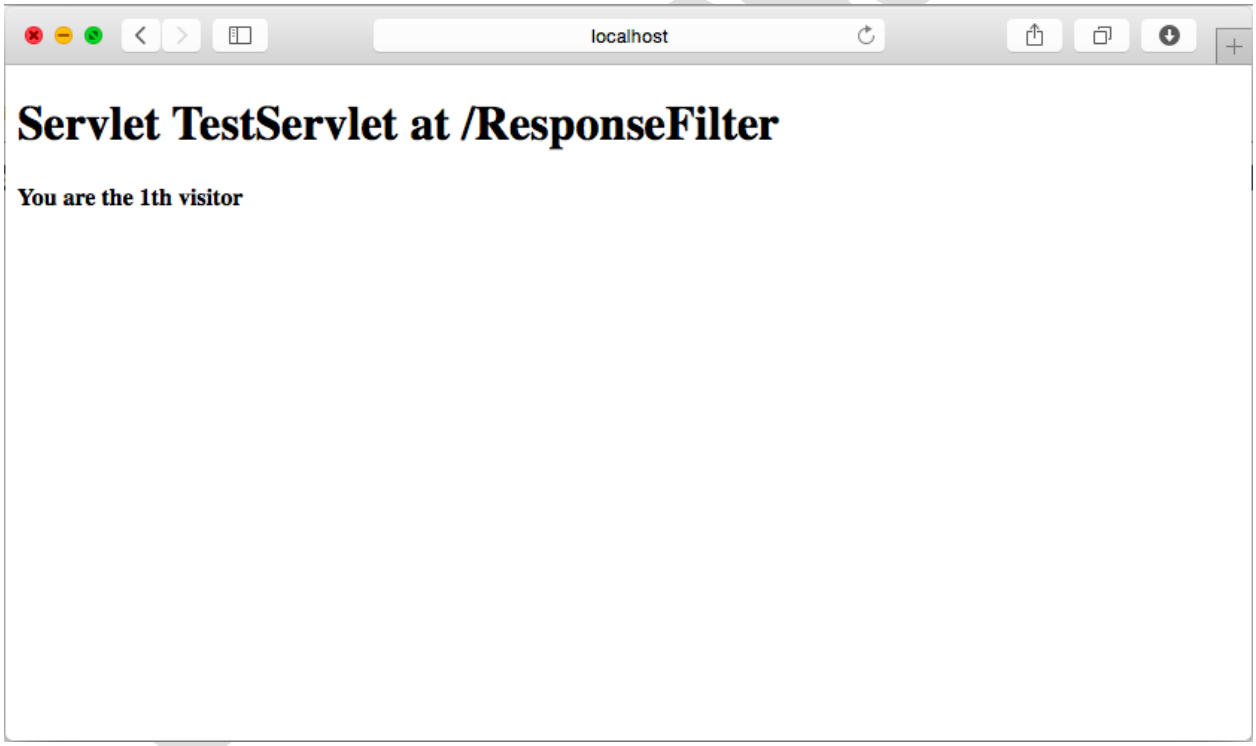
```java
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.DispatcherType;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletResponse;
import util.FilterResponseWrapper;

@WebFilter(filterName = "VisitorCounterFilter",
        urlPatterns = {"/SimpleServlet"},
        dispatcherTypes = {
            DispatcherType.REQUEST,
            DispatcherType.FORWARD,
            DispatcherType.INCLUDE})
public class VisitorCounterFilter implements Filter {

    private FilterConfig filterConfig = null;

    private ServletContext context;

    public VisitorCounterFilter() {
    }

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        this.filterConfig = filterConfig;
        this.context = filterConfig.getServletContext();
        this.context.setAttribute("visitorCount", new Integer(0));
    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws IOException, ServletException {
        int i = ((Integer)
this.context.getAttribute("visitorCount")).intValue();
        i++;
        this.context.setAttribute("visitorCount", new Integer(i));

        FilterResponseWrapper res = new
FilterResponseWrapper((HttpServletResponse) response);
        chain.doFilter(request, res);

        PrintWriter out = response.getWriter();
        out.println(res.toString());
        out.println("<b>You are the " + i + "visitor </b>");
        out.close();
```

```
    }

    @Override
    public void destroy() {
        throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.
    }

    public void log(String msg) {
        filterConfig.getServletContext().log(msg);
    }

}
```

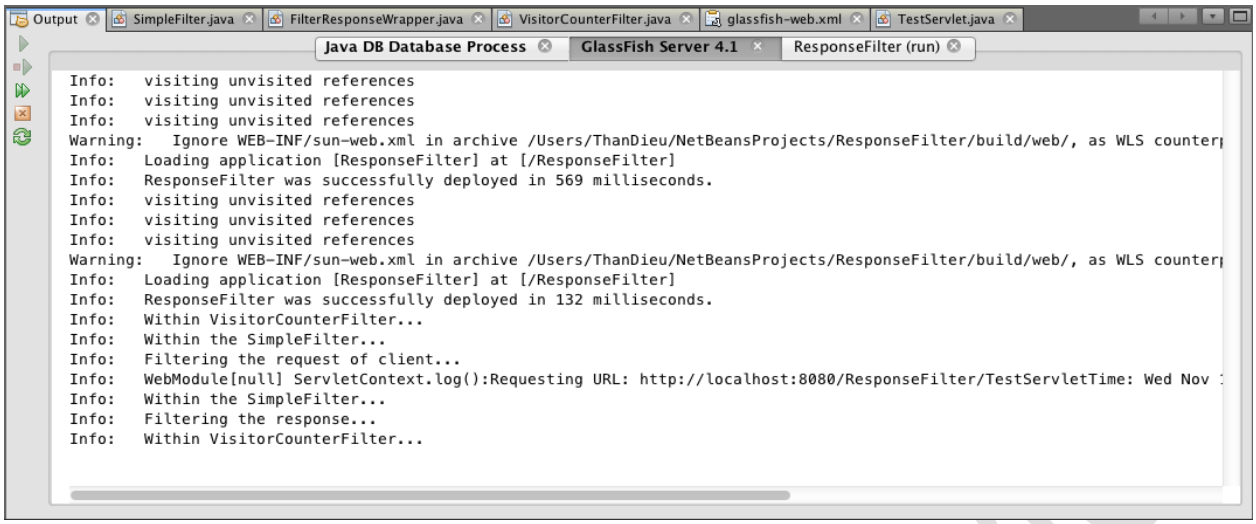
- Thiết lập thông số để "**SimpleServlet**" được chạy trước:
- Click chuột phải vào project chọn "**Properties / Run / Relative URL**". Điền vào giá trị:

  /SimpleServlet

Chạy ứng dụng, xem log trên server.



Log Server



HẾT