

## Lab 06

# Introduction to Asynchronous Servlet

### Mục tiêu

- Asynchronous Servlet

### Phần I Bài tập step by step

---

#### Bài 1.1

##### Mục tiêu:

- Hiểu hoạt động của Async Servlet
- Cách implement một servlet chạy ở chế độ không đồng bộ

##### Nhắc lại:

Async Servlet được sử dụng để tối ưu nguồn tài nguyên của Web Applications, cho phép chúng ta có thể tạo riêng thread để điều khiển request và viết các response nhằm giảm thời gian chờ. Nội dung sử dụng Async Servlet:

- Request cần nhiều thời gian để xử lý và xây dựng response trả lại cho client
- Cần streaming response hoặc response đợi các xử lý dữ liệu từ các luồng khác phải được hoàn thành do đó nó ở trong trạng thái đợi xử lý

Async Servlet sẽ trì hoãn các response cần nhiều thời gian để xử lý và xây dựng một thread khác để viết các response đó.

Trong bài thực hành này, chúng ta sẽ xây dựng Async Servlet để xử lý request của client chiếm nhiều thời gian để xử lý.

#### STEP 1: Tạo ứng dụng web tên AsyncListener

New Project -> Java Web -> Web Application: AsyncListener

Server: GlassFish

Java EE Version 7 -> Finish

## STEP 2: Tạo tình huống xử lý request chiếm nhiều thời gian

Tạo package servlet -> New Servlet: LongRunningServlet

Trong servlet này, chúng ta đơn giản cho Thread sleep một khoảng thời gian (10000ms). Sau đó trả về response cho client.

```
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;

@WebServlet(urlPatterns = {"/LongRunningServlet"})
public class LongRunningServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        long startTime = System.currentTimeMillis ();
        System.out.println("LongRunningServlet Start::Name="
            + Thread.currentThread().getName() + "::ID="
            + Thread.currentThread().getId());

        try{
            Thread.sleep(8000);
        }
        catch(InterruptedException ex){
            ex.printStackTrace();
        }

        PrintWriter out = response.getWriter();
        long endTime = System.currentTimeMillis ();
        out.write("Work completed. Time elapsed: " + (endTime - startTime));
        System.out.println("LongRunningServlet Start::Name="
            + Thread.currentThread().getName() + "::ID="
            + Thread.currentThread().getId() + "::Time Taken="
            + (endTime - startTime) + " ms");

    }
}
```

Gõ vào trình duyệt:

http://localhost:8080/AsyncListener/ LongRunningServlet

Log trên server:

Info: LongRunningServlet Start::Name=http-listener-1(5)::ID=122

Info: LongRunningServlet Start::Name=http-listener-1(5)::ID=122::Time  
Taken=8003 ms

Thread ID 122 được tạo ra để xử lý cho request này, thời gian xử lý là 8003ms.

Đặt ra tình huống là, http-thread-pool của server chỉ cấp maximum thread là 5.

**Thread Pools**

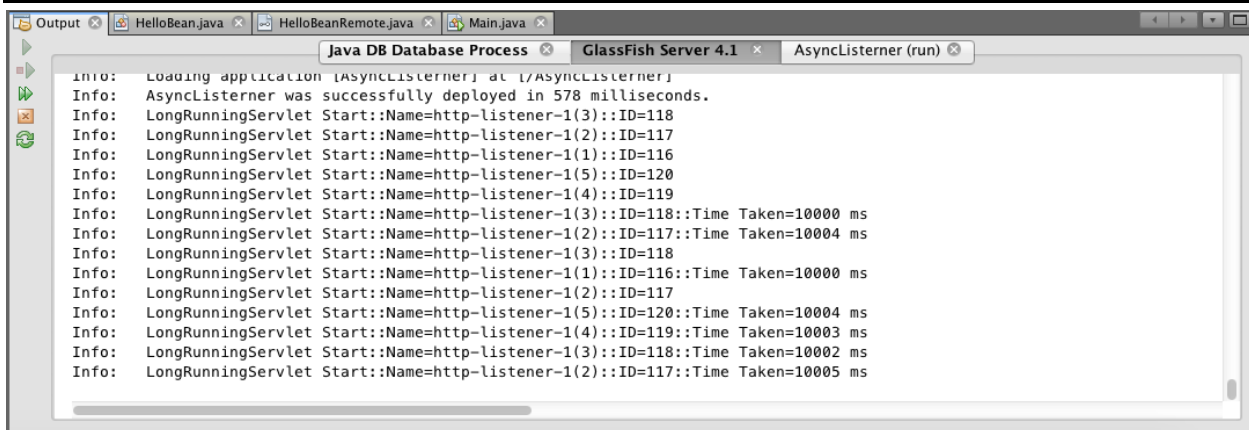
Use thread pools to limit a service to a specific number of concurrent threads.

Configuration Name: server-config

Thread Pools (3)

Select	Thread Pool ID	Max Thread Pool Size	Min Thread Pool Size	Max Queue Size	Idle Thread Timeout
<input type="checkbox"/>	admin-thread-pool	50	5	256	900
<input type="checkbox"/>	http-thread-pool	5	5	4096	900
<input type="checkbox"/>	thread-pool-1	200	5	4096	900

Bây giờ nếu chúng ta tạo ra 7 request tới server (gần như đồng thời). Log trên server:



Lần lượt, 5 thread sẽ được sử dụng. Chỉ khi nào 1 thread làm xong nhiệm vụ, nó mới được giải phóng và phục vụ request thứ 6 (Thread ID 118). Sau đó là thread ID 117 phục vụ request thứ 7. Điều này có nghĩa là trong khoảng thời gian cả 5 thread cùng hoạt động server rơi vào trạng thái không thể đáp ứng (non-responsive).

Async Servlet được sử dụng trong những tình huống đó, hạn chế xảy ra tình trạng non-responsive của server.

## STEP 2: Tạo async servlet

Tạo package servlet.async -> Tạo servlet **AsyncLongRunningServlet**.

```
package servlet.async;

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

@WebServlet(
    name = "AsyncLongRunningServlet",
    urlPatterns = {"/AsyncLongRunningServlet"},
    asyncSupported = true)
public class AsyncLongRunningServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        final long startTime = System.currentTimeMillis();
```

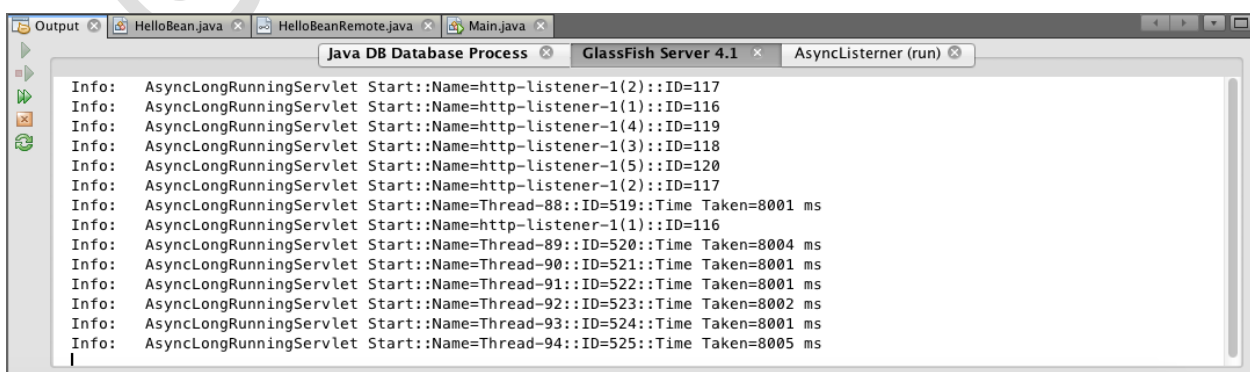
```
//Inform server that this servlet is run in async mode
final AsyncContext asyncContext = request.startAsync(request,
response);
new Thread() {
    @Override
    public void run() {
        try {
            ServletResponse response = asyncContext.getResponse();
            response.setContentType("text/plain");
            PrintWriter out = response.getWriter();
            Thread.sleep(8000);
            out.print("Work completed. Time elapsed: " + (System.
currentTimeMillis () - startTime));
            out.flush();
            asyncContext.complete();
        } catch (IOException | InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
}.start();
}
```

Cùng một kịch bản với LongRunningServlet, tuy nhiên container sẽ được thông báo rằng servlet AsyncLongRunningServlet sẽ chạy ở chế độ không đồng bộ nhờ thuộc tính `asyncSupported = true`.

Chế độ không đồng bộ sẽ được start thông qua: `AsyncContext asyncContext = request.startAsync(request, response);`

Một thread mới sẽ được tạo ra và được chuyển giao toàn bộ hoạt động của servlet (`Thread.sleep(8000)`). Khi kết thúc cần phải thông báo cho server: `asyncContext.complete();`

Tạo ra 7 request đồng thời tới server để xem khả năng đáp ứng của server.



```
Output
HelloBean.java x HelloBeanRemote.java x Main.java x
Java DB Database Process x GlassFish Server 4.1 x AsyncListener (run) x
Info: AsyncLongRunningServlet Start::Name=http-listener-1(2)::ID=117
Info: AsyncLongRunningServlet Start::Name=http-listener-1(1)::ID=116
Info: AsyncLongRunningServlet Start::Name=http-listener-1(4)::ID=119
Info: AsyncLongRunningServlet Start::Name=http-listener-1(3)::ID=118
Info: AsyncLongRunningServlet Start::Name=http-listener-1(5)::ID=120
Info: AsyncLongRunningServlet Start::Name=http-listener-1(2)::ID=117
Info: AsyncLongRunningServlet Start::Name=Thread-88::ID=519::Time Taken=8001 ms
Info: AsyncLongRunningServlet Start::Name=http-listener-1(1)::ID=116
Info: AsyncLongRunningServlet Start::Name=Thread-89::ID=520::Time Taken=8004 ms
Info: AsyncLongRunningServlet Start::Name=Thread-90::ID=521::Time Taken=8001 ms
Info: AsyncLongRunningServlet Start::Name=Thread-91::ID=522::Time Taken=8001 ms
Info: AsyncLongRunningServlet Start::Name=Thread-92::ID=523::Time Taken=8002 ms
Info: AsyncLongRunningServlet Start::Name=Thread-93::ID=524::Time Taken=8001 ms
Info: AsyncLongRunningServlet Start::Name=Thread-94::ID=525::Time Taken=8005 ms
```

Sau khi 5 thread đều đã bị chiếm dụng (lần lượt là 117, 116, 119, 118, 120), mặc dù response của 5 request đó chưa được tạo ra (chưa có log Time Taken), request thứ 6 vẫn được đáp ứng bởi server. Lí do là vì, sau khi chuyển giao công việc cho 1 thread khác (chạy ngầm) thì thread 117 được giải phóng và sẵn sàng phục vụ. Async Servlet giúp làm giảm thiểu khả năng rơi vào tình trạng non-responsive của server.

Qua bài này, chúng ta đã hiểu được nguyên lý hoạt động của Async Servlet.

## Bài 1.2

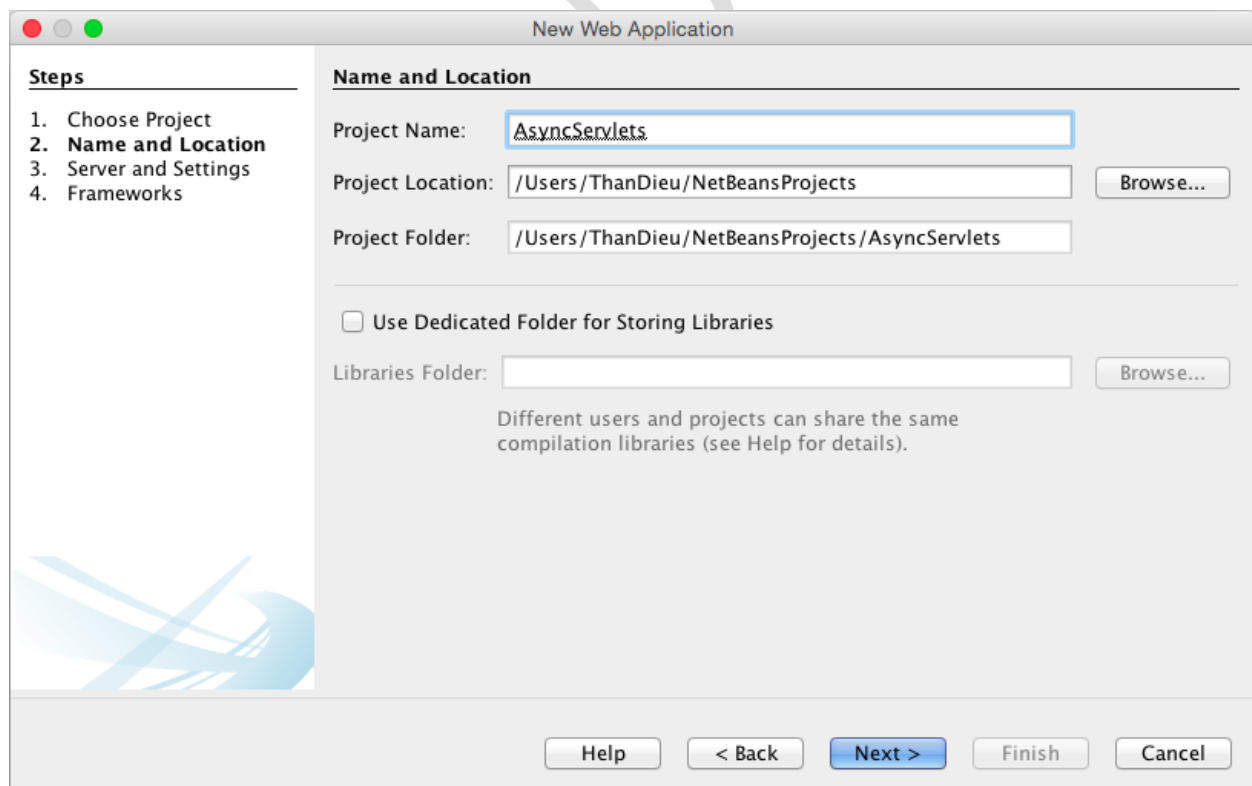
Async Servlet còn được sử dụng để server push message không đồng bộ đến client.

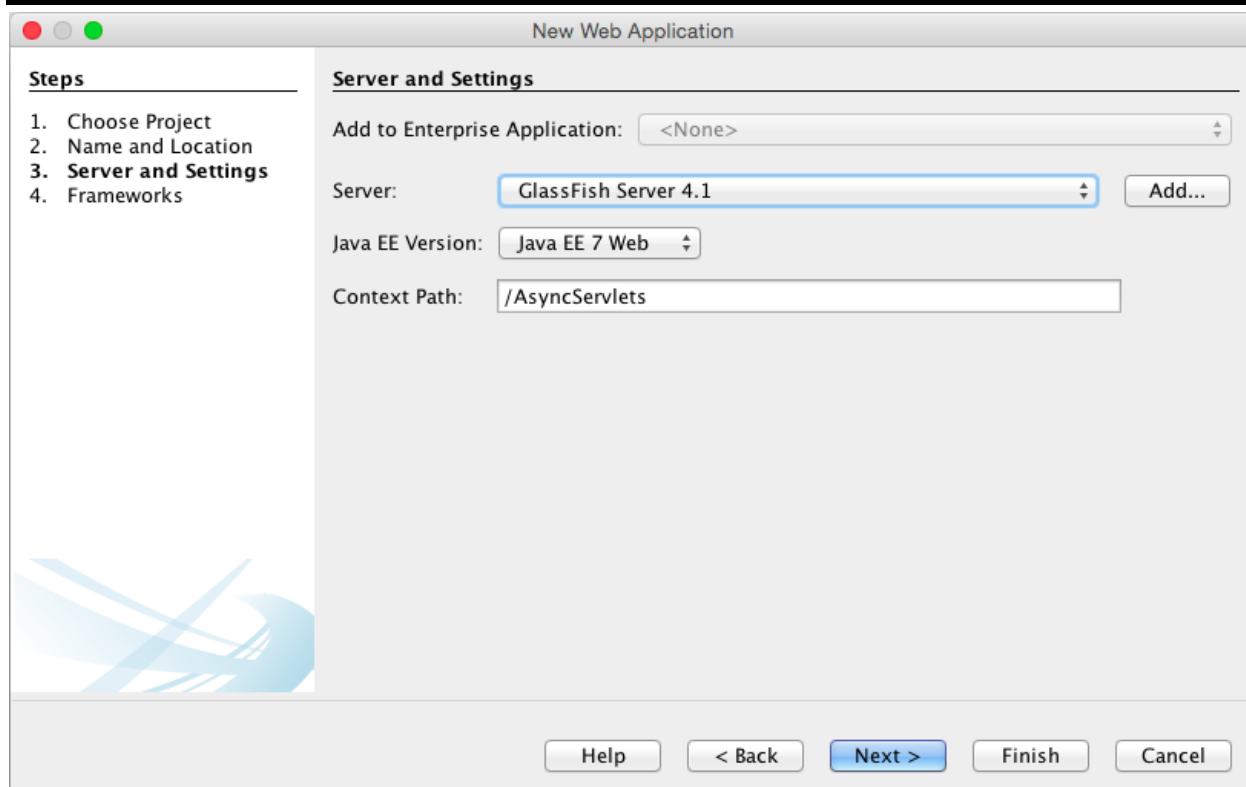
Trong bài thực hành này chúng ta sẽ xây dựng ứng dụng Web Push Notifications

### STEP 1: Tạo ứng dụng web với netbeans

File -> New Project -> Java Web -> Web Application

Project Name: AsyncServlets





Finish

## STEP 2: Tạo trang jsp view gọi đến servlet

Xoá trang index.html được tạo sẵn bởi Netbeans (nếu có).

Tạo trang index.jsp, thêm code như sau:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h1>SHOUT-OUT!</h1>
    <form method="post" action="ShoutServlet">
      <table>
        <tr>
          <td>Your name:</td>
          <td><input type="text" id="name" name="name"/></td>
        </tr>
        <tr>
          <td>Your shout:</td>
          <td><input type="text" id="message" name="message">
        </tr>
      </table>
    </form>
  </body>
</html>
```

```
</tr>
<tr>
    <td><input type="submit" value="SHOUT" /></td>
</tr>
</table>
</form>
<h2> Current Shouts </h2>
<div id="content">
    <% if (application.getAttribute("messages") != null) {%>
    <%= application.getAttribute("messages")%>
    <% }%>
</div>
</body>
</html>
```

## STEP 2: Tạo Servlet ShoutServlet

Tạo mới package: servlet

Tạo servlet **ShoutServlet** trong package servlet

New Servlet

**Steps**

1. Choose File Type
2. **Name and Location**
3. Configure Servlet Deployment

**Name and Location**

Class Name:

Project:

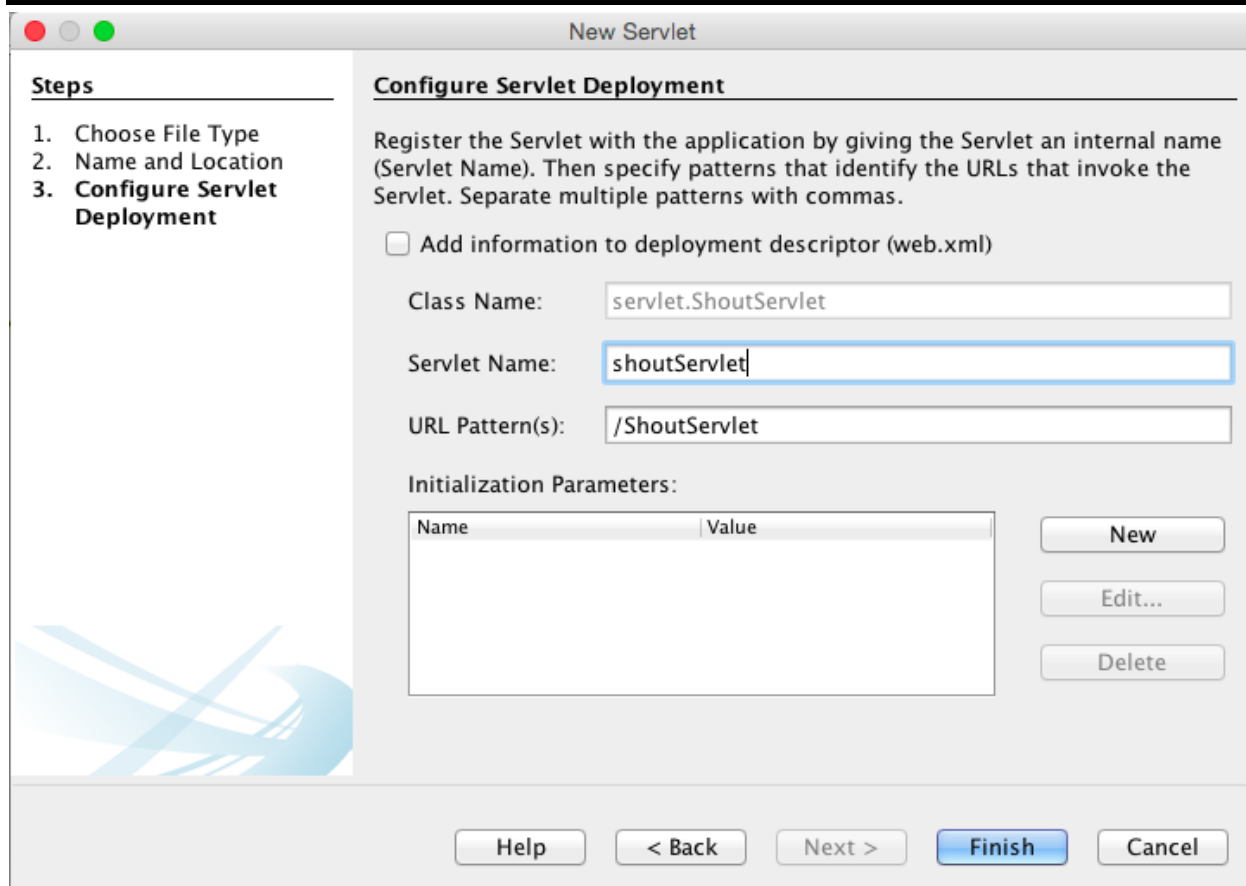
Location:

Package:

Created File:

Help < Back **Next >** Finish Cancel





**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 Name:

URL Pattern(s):

Initialization Parameters:

Name	Value
------	-------

New Edit... Delete

Help < Back Next > Finish Cancel

```
package servlet;

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

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

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        String name = request.getParameter("name");
        String message = request.getParameter("message");
        String htmlMessage = "<p><b>" + name + "</b><br/>" + message + "</p>";
        ServletContext sc = request.getServletContext();
        if (sc.getAttribute("messages") == null) {
            sc.setAttribute("messages", htmlMessage);
        } else {
            String currentMessages = (String) sc.getAttribute("messages");
            sc.setAttribute("messages", htmlMessage + currentMessages);
        }
    }
}
```

```
}  
    response.sendRedirect("index.jsp");  
}  
  
}
```

**STEP 3:** Test ứng dụng trước khi tạo Async Servlet

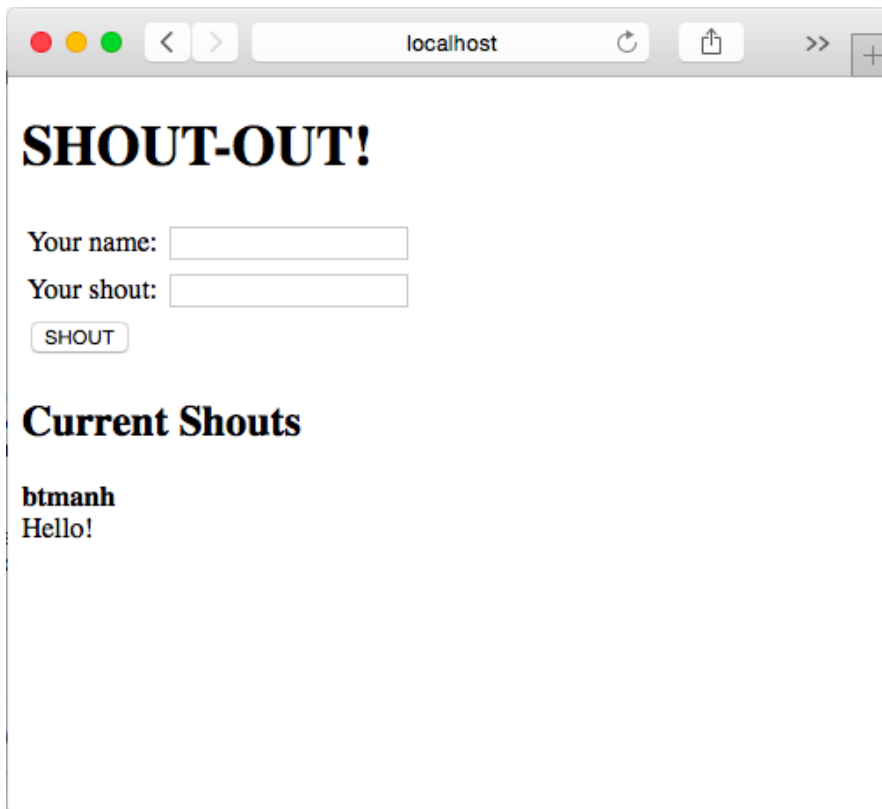
Project -> Clean and Build -> Run

**SHOUT-OUT!**

Your name:

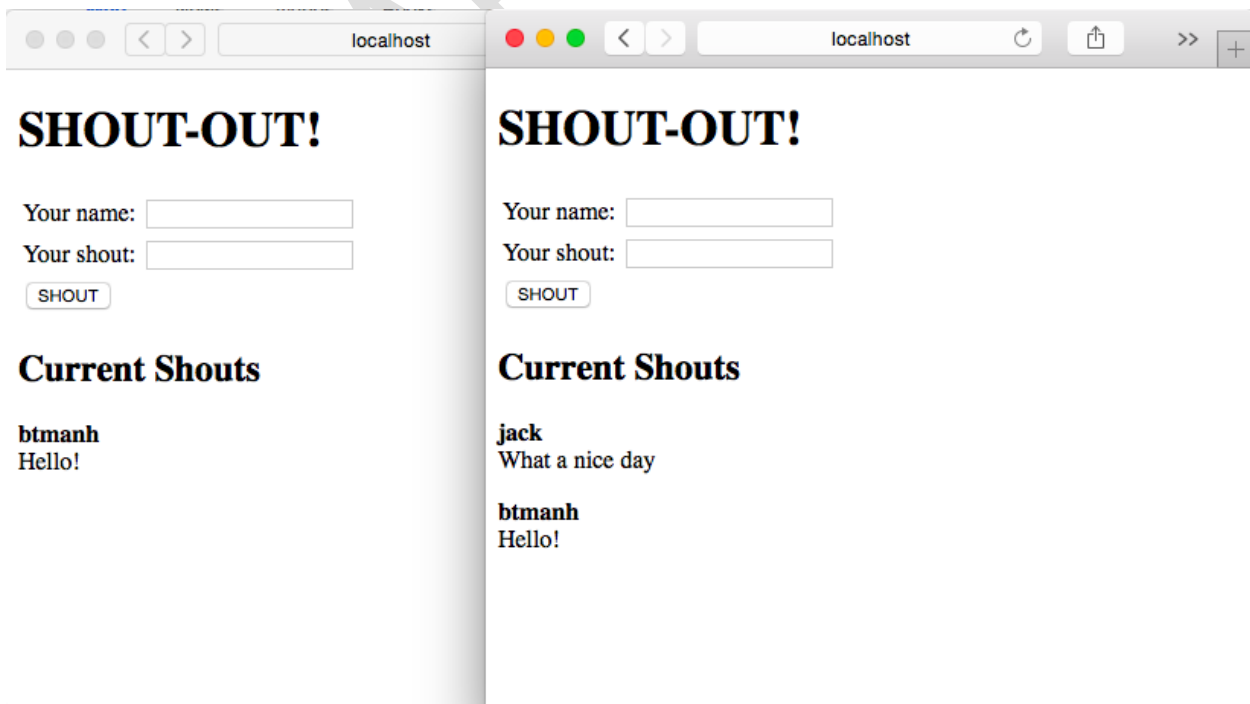
Your shout:

**Current Shouts**



Mở trình duyệt mới, gõ lại đường link tới ứng dụng (<http://localhost:8080/AsyncServlets/>)

Cập nhật thêm thông tin mới, chúng ta thấy chỉ một bên trang được update shout information



**STEP 4: Chèn thêm ajax code sử dụng Async Servlet để refresh trang, cập nhật thông tin**

Chèn thêm đoạn script sau vào trang index.jsp

```
<script>
    function postMessage() {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.open("post", "ShoutServlet?t="+new Date(), false);
        xmlhttp.setRequestHeader("Content-Type", "application/x-www-
form-urlencoded");
        var nameText = escape(document.getElementById("name").value);
        var messageText =
escape(document.getElementById("message").value);
        document.getElementById("message").value = "";
        xmlhttp.send("name="+nameText+"&message="+messageText);
    }
    var messagesWaiting = false;
    function getMessages(){
        if(!messagesWaiting){
            messagesWaiting = true;
            var xmlhttp = new XMLHttpRequest();
            xmlhttp.onreadystatechange=function(){
                if (xmlhttp.readyState==4 && xmlhttp.status==200) {
                    messagesWaiting = false;
                    var contentElement =
document.getElementById("content");
                    contentElement.innerHTML = xmlhttp.responseText +
contentElement.innerHTML;
                }
            }
            xmlhttp.open("GET", "shoutServlet?t="+new Date(), true);
            xmlhttp.send();
        }
    }
    setInterval(getMessages, 1000);
</script>
```

Đoạn script này sẽ tiếp tục request tới ShoutServlet khi có message mới, tạo response và gắn vào content div trên trang index.jsp.

Sửa lại button code như sau:

```
<input type="button" onclick="postMessage();" value="SHOUT" />
```

Tiếp theo chúng ta sẽ sửa lại ShoutServlet:

```
package servlet;

import java.io.IOException;
```

```
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import javax.servlet.AsyncContext;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(urlPatterns = {"/ShoutServlet"}, asyncSupported=true)
public class ShoutServlet extends HttpServlet {
    private List<AsyncContext> contexts = new LinkedList<>();

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        final AsyncContext asyncContext = request.startAsync(request,
response);
        asyncContext.setTimeout(10 * 60 * 1000);
        contexts.add(asyncContext);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        List<AsyncContext> asyncContexts = new ArrayList<>(this.contexts);
        this.contexts.clear();
        String name = request.getParameter("name");
        String message = request.getParameter("message");
        String htmlMessage = "<p><b>" + name + "</b><br/>" + message + "</p>";
        ServletContext sc = request.getServletContext();
        if (sc.getAttribute("messages") == null) {
            sc.setAttribute("messages", htmlMessage);
        } else {
            String currentMessages = (String) sc.getAttribute("messages");
            sc.setAttribute("messages", htmlMessage + currentMessages);
        }
        for (AsyncContext asyncContext : asyncContexts) {
            try (PrintWriter writer = asyncContext.getResponse().getWriter())
            {
                writer.println(htmlMessage);
                writer.flush();
                asyncContext.complete();
            } catch (Exception ex) {
            }
        }
    }
}
```

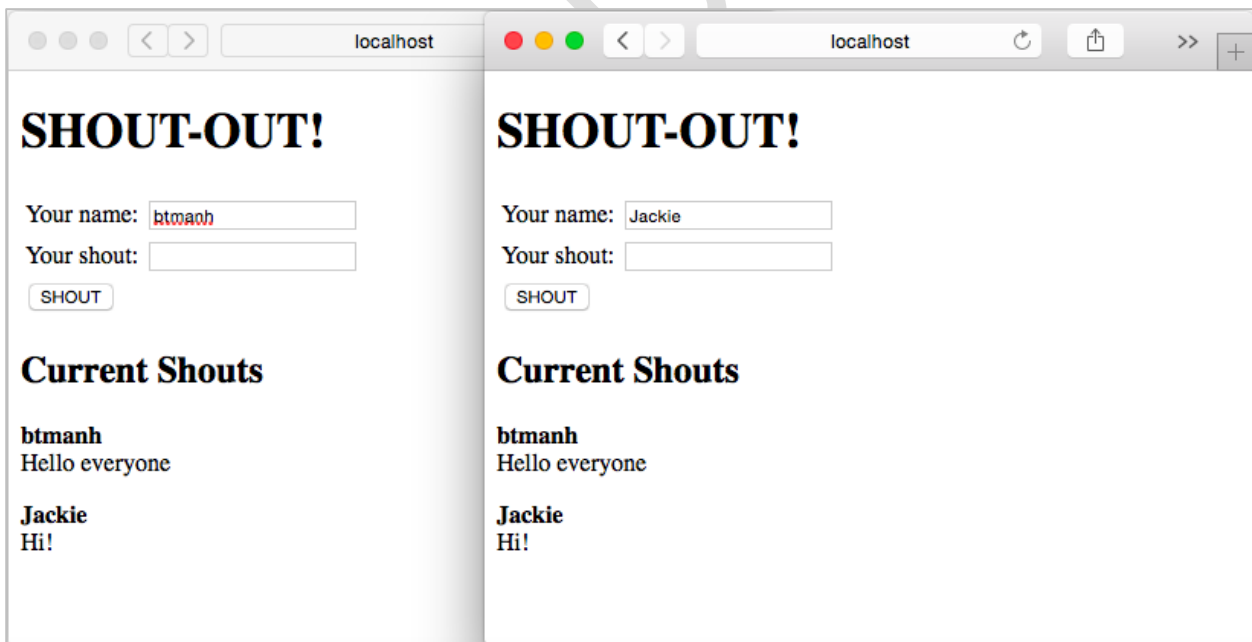
```
}
```

Khi servlet nhận được một “get request”, AsyncContext sẽ được khởi tạo bằng cách gọi: `request.startAsync(request, response)`. Web Container sẽ được thông báo phải free thread khi request kết thúc và mở connection để các thread khác có thể ghi lên response và kết thúc connection.

Tiếp theo chúng ta tạo một list các AsyncContext, add các asyncContext được tạo ra khi có request get vào list. Trong phương thức `doPost`, chúng ta sẽ get về tất cả các waiting context này.

Trong phương thức `doPost`, chúng ta truy cập list các AsyncContext (tạo safe copy của list này, clear list cũ để đảm bảo các pending request không bị notified hai lần. Đối với mỗi asyncContext trong list, chúng ta sẽ ghi các message vào response và kết thúc connection bằng `asyncContext.complete()`).

#### STEP 5: Test



## Bài 1.2: Thực hành với Async Listener

Mục tiêu:

- Trong bài tập này, chúng ta tạo async servlet, thực hành với async listener và thread pool.

### STEP 1: Tạo mới project web tên: AsyncListener

**Steps**

1. Choose Project
2. **Name and Location**
3. Server and Settings
4. Frameworks

**Name and Location**

Project Name:

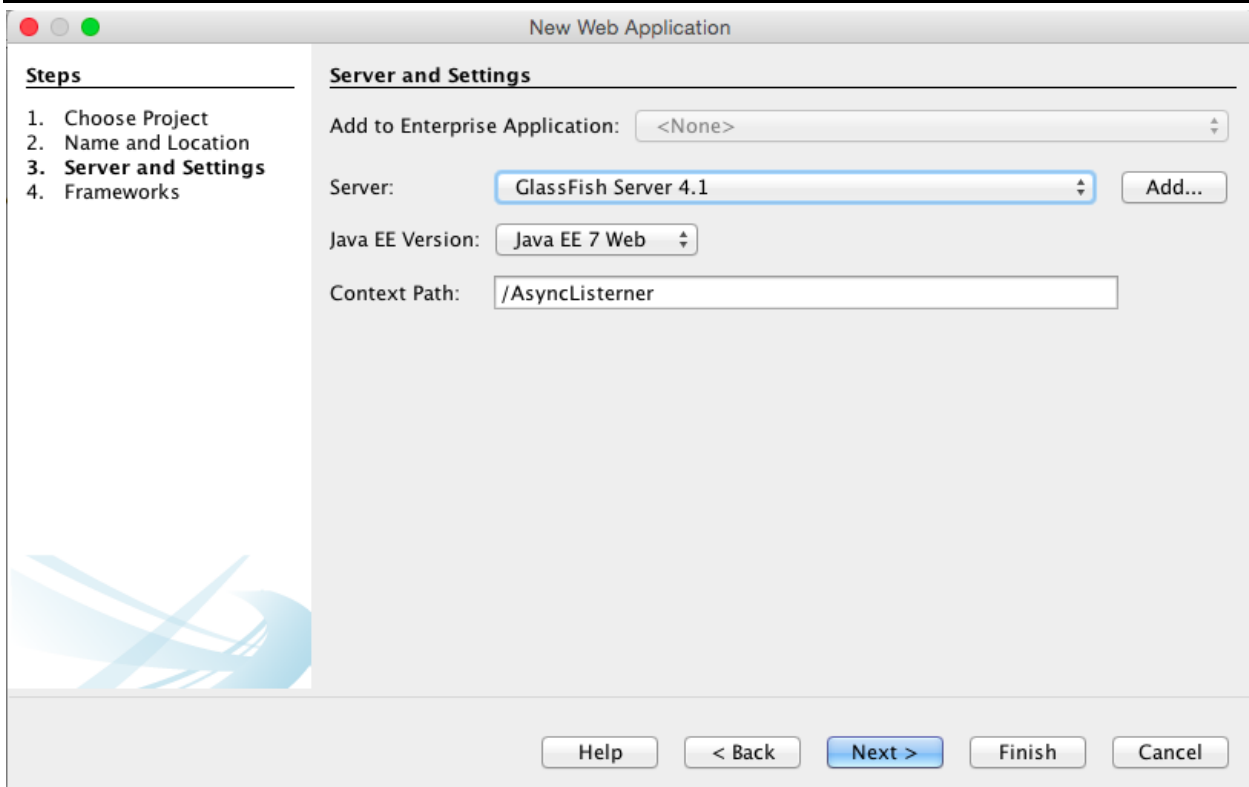
Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).



STEP 2: Tạo Servlet: AppExceptionHandlerServlet

**HẾT**