**­Q56 - Asynchronous Processing in Servlets**

Considering the following asynchronous servlet code, choose which statements are true after a GET request is made:

package com.nullhaus;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/foo/\*", name="NullServlet")

public class NullServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) {

System.out.println("I'm inside!");

final AsyncContext ac = req.startAsync();

ac.start(new Runnable() {

public void run() {

System.out.println("\*\*\* I'm an async thread!");

ac.complete();

}

});

System.out.println("I'm leaving! Bye!");

}

}

1. This code compiles and runs fine
2. The modifier "final" in final AsyncContext ac = req.startAsync(); is not necessary and can be safely removed
3. The guaranteed order of texts printed in the console/log file is: I'm inside!, \*\*\* I'm an async thread!, I'm leaving! Bye!
4. There is no HttpServletRequest#startAsync() method - there is only a HttpServletRequest#startAsync(ServletRequest, ServletResponse) method
5. A runtime exception will be thrown when accessing this servlet
6. This code doesn't compile

**Q57 - Asynchronous Processing in Servlets [2]**

Considering the following asynchronous servlet code, choose which statements are true after a GET request is made:

package com.nullhaus;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/foo/\*", name="NullServlet", asyncSupported = true)

public class NullServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) {

System.out.println("I'm inside!");

final AsyncContext ac = req.startAsync();

ac.start(new Runnable() {

public void run() {

System.out.println("\*\*\* I'm an async thread!");

ac.complete();

}

});

System.out.println("I'm leaving! Bye!");

}

}

1. This code compiles and runs fine
2. There is no "asyncSupported" attribute of @WebServlet
3. The modifier "final" in final AsyncContext ac = req.startAsync(); is not necessary and can be safely removed
4. The guaranteed order of texts printed in the console/log file is: I'm inside!, \*\*\* I'm an async thread!, I'm leaving! Bye!
5. There is no HttpServletRequest#startAsync() method - there is only a HttpServletRequest#startAsync(ServletRequest, ServletResponse) method
6. A runtime exception will be thrown when accessing this servlet
7. This code doesn't compile

**Q58 - Asynchronous Processing in Servlets [3]**

Considering the following asynchronous servlet code, choose which statements are true after a GET request is made:

package com.nullhaus;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/foo/\*", name = "NullServlet", asyncSupported = true)

public class NullServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) {

final AsyncContext ac = req.startAsync();

ac.start(new Runnable() {

public void run() {

ac.dispatch("/page.html");

ac.complete();

}

});

}

}

1. This code compiles
2. The content of "/page.html" will be served as a response
3. A runtime exception will be thrown when accessing this servlet
4. This code doesn't compile

**Q59 - Asynchronous Processing in Servlets [4]**

Considering the following asynchronous servlets code, choose which statements are true after a GET request to the NullServlet is made:

**com.nullhaus.NullServlet**

package com.nullhaus;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/foo/\*", name = "NullServlet", asyncSupported = true)

public class NullServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {

resp.getWriter().println("Howdy from NullServlet1!");

final AsyncContext ac = req.startAsync();

ac.start(new Runnable() {

public void run() {

ac.dispatch("/baz");

}

});

}

}

**com.nullhaus.NullServlet2**

package com.nullhaus;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/baz/\*", name = "NullServle2", asyncSupported = false)

public class NullServlet2 extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {

resp.getWriter().println("Howdy from NullServlet2!");

}

}

1. This code compiles
2. The "Howdy from NullServlet2" will be included in the response
3. The "Howdy from NullServlet1" will be included in the response
4. A runtime exception will be thrown when accessing this servlet
5. This code doesn't compile

**Q60 - Asynchronous Processing in Servlets [5]**

Considering the following Asynchronous Servlets code, choose which statements are true after a GET request to the NullServlet2 is made:

package com.nullhaus;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/foo/\*", name = "NullServlet", asyncSupported = true)

public class NullServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp)

throws IOException {

resp.getWriter().println("Howdy from NullServlet1!");

}

}

package com.nullhaus;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/baz/\*", name = "NullServle2", asyncSupported = false)

public class NullServlet2 extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {

resp.getWriter().println("Howdy from NullServlet2!");

req.getRequestDispatcher("/foo").forward(req, resp);

}

}

1. This code compiles
2. The "Howdy from NullServlet2" will be included in the response
3. The "Howdy from NullServlet1" will be included in the response
4. A runtime exception will be thrown when accessing this servlet
5. This code doesn't compile

**Q61 - Asynchronous Processing in Servlets [6]**

Considering the following Asynchronous Servlets code, choose which statements are true after a GET request to the NullServlet2 is made:

**com.nullhaus.NullServlet**

package com.nullhaus;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/foo/\*", name = "NullServlet", asyncSupported = true)

public class NullServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {

resp.getWriter().println("Howdy from NullServlet1!");

final AsyncContext ac = req.startAsync();

ac.start(new Runnable() {

public void run() {

System.out.println("Async!");

ac.complete();

}

});

}

}

**com.nullhaus.NullServlet2**

package com.nullhaus;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/baz/\*", name="NullServle2", asyncSupported=false)

public class NullServlet2 extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {

resp.getWriter().println("Howdy from NullServlet2!");

req.getRequestDispatcher("/foo").forward(req, resp);

}

}

1. This code compiles
2. The "Howdy from NullServlet2" will be included in the response
3. The "Howdy from NullServlet1" will be included in the response
4. The "Async!" will be included in the response
5. A runtime exception will be thrown when accessing this servlet
6. This code doesn't compile

**Q62 - Asynchronous Processing in Servlets [7]**

Considering the following Asynchronous Servlet code, choose which statements are true after a GET request to the servlet is made:

package com.nullhaus;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/foo/\*", name="NullServlet", asyncSupported=true)

public class NullServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {

req.setAttribute("Hello", "World");

final AsyncContext ac = req.startAsync();

ac.setTimeout(-2);

ac.start(new Runnable() {

public void run() {

String att = (String)ac.getRequest().getAttribute("Hello");

try {

PrintWriter pw = ac.getResponse().getWriter();

pw.println("Async! Value of Hello is: " + att);

} catch (IOException e) {

e.printStackTrace();

}

ac.complete();

}

});

}

}

1. This code compiles
2. The "Async! Value of Hello is: null" will be included in the response
3. The "Async! Value of Hello is: World" will be included in the response
4. The asynchronous operation will be timed out after the server default timeout value
5. The asynchronous operation will be never timed out
6. A runtime exception will be thrown when accessing this servlet
7. This code doesn't compile

**Q63 - Asynchronous Processing in Servlets [8]**

Considering the following Asynchronous Servlets code, choose which statements are true after a GET request to the NullServlet is made:

**com.nullhaus.NullServlet**

package com.nullhaus;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/foo/\*", name="NullServlet", asyncSupported=true)

public class NullServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {

if (req.getDispatcherType() == DispatcherType.REQUEST) {

req.getRequestDispatcher("/baz").forward(req, resp);

} else if (req.getDispatcherType() == DispatcherType.ASYNC) {

String hello = (String)req.getAttribute("Hello");

resp.getWriter().println("Phew, that was a ride!");

resp.getWriter().println("Value of Hello is: " + hello);

}

}

}

**com.nullhaus.NullServlet2**

package com.nullhaus;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/baz/\*", name="NullServlet2", asyncSupported=true)

public class NullServlet2 extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {

req.setAttribute("Hello", "World");

final AsyncContext ac = req.startAsync();

ac.dispatch();

}

}

1. This code compiles
2. The "Value of Hello is: null" will be included in the response
3. The "Value of Hello is: World" will be included in the response
4. The infinite loop dispatch-loop will be created
5. The request will be served fine, but no text will be included in the response
6. A runtime exception will be thrown when accessing this servlet
7. This code doesn't compile

**Q64 - Asynchronous Processing in Servlets [9]**

Considering the following Asynchronous Servlets code, choose which statements are true after a GET request to the NullServlet is made:

**com.nullhaus.NullServlet**

package com.nullhaus;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/foo/\*", name="NullServlet", asyncSupported=true)

public class NullServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {

if (req.getDispatcherType() == DispatcherType.REQUEST) {

AsyncContext ac = req.startAsync();

ac.dispatch("/baz");

} else if (req.getDispatcherType() == DispatcherType.ASYNC) {

resp.getWriter().println("Shotgun!");

}

}

}

**com.nullhaus.NullServlet2**

package com.nullhaus;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/baz/\*", name="NullServlet2", asyncSupported=true)

public class NullServlet2 extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {

AsyncContext ac = req.getAsyncContext();

ac.dispatch("/foo");

}

}

1. This code compiles
2. The "Shotgun!" will be included in the response
3. The inifite loop dispatch-loop will be created
4. The request will be served fine, but no text will be included in the response
5. A runtime exception will be thrown when accessing this servlet
6. This code doesn't compile

**Q65 - Asynchronous Processing in Servlets [10]**

Considering the following Asynchronous Servlets code, choose which statements are true after a GET request to the NullServlet is made:

**com.nullhaus.NullServlet**

package com.nullhaus;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/foo/\*", name="NullServlet", asyncSupported=true)

public class NullServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {

if (req.getDispatcherType() == DispatcherType.REQUEST) {

AsyncContext ac = req.startAsync();

ac.dispatch("/baz");

} else if (req.getDispatcherType() == DispatcherType.ASYNC) {

resp.getWriter().println("Shotgun!");

}

}

}

**com.nullhaus.NullServlet2**

package com.nullhaus;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/baz/\*", name="NullServlet2", asyncSupported=true)

public class NullServlet2 extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {

AsyncContext ac = req.startAsync();

ac.dispatch("/foo");

}

}

1. This code compiles
2. The "Shotgun!" will be included in the response
3. The inifite loop dispatch-loop will be created
4. The request will be served fine, but no text will be included in the response
5. A runtime exception will be thrown when accessing this servlet
6. This code doesn't compile

**Q66 - Asynchronous Processing in Servlets [11]**

Assume the AsyncListener is defined as follows:

**com.nullhaus.MyListener**

package com.nullhaus;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebListener

public class MyListener implements AsyncListener {

public void onComplete(AsyncEvent event) {

System.out.println("#Async listener [onComplete]");

}

public void onError(AsyncEvent event) {

System.out.println("#Async listener [onError]");

}

public void onStartAsync(AsyncEvent event) {

System.out.println("#Async listener [onStartAsync]");

}

public void onTimeout(AsyncEvent event) {

System.out.println("#Async listener [onTimeout]");

}

}

What will be the result of the **first** GET request to the following servlet:

package com.nullhaus;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.\*;

@WebServlet(urlPatterns = "/foo/\*", name="NullServlet", asyncSupported=true)

public class NullServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException {

final AsyncContext ac = req.startAsync();

try {

Class lClass = Class.forName("com.nullhaus.MyListener");

AsyncListener al = ac.createListener(lClass);

ac.addListener(al);

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

ac.start(new Thread() {

public void run() {

ac.complete();

}

});

}

}

1. This code compiles
2. The "#Async listener [onTimeout]" message will be to the console/log file
3. The "#Async listener [onStartAsync]" message will be to the console/log file
4. The "#Async listener [onError]" message will be to the console/log file
5. The "#Async listener [onComplete]" message will be to the console/log file
6. A runtime exception will be thrown when accessing this servlet
7. This code doesn't compile