PARTE J
PRIMO ESEMPIO DI SERVLET

Modulo 1
Overview Servlet

Definizione di servlet

A servlet is a Java program that runs on a Web server.

It is similar to an applet, but is processed on the server rather than a client's machine.

Servlets are often run when the user clicks a link, submits a form, or performs another type of action on a website.

Caratteristiche di una servlet

- **Discende da una classe di tipo:**
  - javax.servlet.GenericServlet
  - javax.servlet.http.HttpServlet

- **Può implementare i metodi**
  - init()
  - destroy()
  - service()

Servlet temporanee e permanenti

- **Le servlet possono essere attivate e interrotte per ogni richiesta client oppure attivate quando viene attivato il Web server e mantenute in vita fino a quando il server non è interrotto**
- **Le servlet temporanee sono caricate on demand e consentono di conservare risorse nel server per le funzioni meno utilizzate**

Servlet temporanee e permanenti

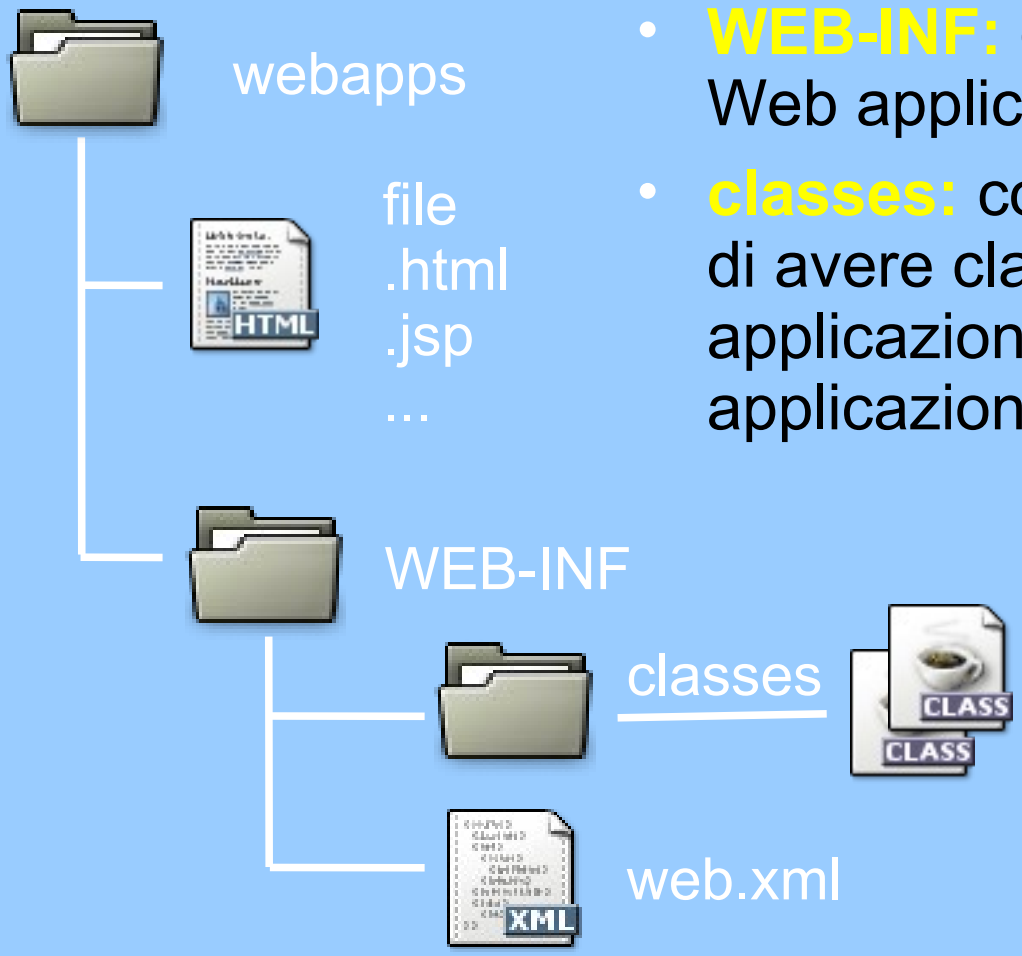
- **Le servlet permanenti servono quando:**
  - costo di attivazione molto alto (connessione a un DBMS)
  - necessità di fornire funzionalità permanenti
  - necessità di rispondere nel modo più veloce possibile alle richieste client
- **Le servlet permanenti restano in memoria anche quando non servono**
  - Aumento della memory footprint del server
  - In generale servlet permanenti sono sconsigliate

Java Servlet API

- **Si trovano in:**
  - Package javax.servlet
  - Package javax.servlet.http

- **Le API supportano 4 categorie di servizi:**
  - Gestione del servlet life cycle
  - Accesso a servlet context
  - Utility classes
  - HTTP-specific support classes

Struttura di una Web application: formato WAR



- **WAR: Web ARrchive**
- **webapps:** top level directory
- **WEB-INF:** directory con dati della specifica Web application
- **classes:** contiene in codice java (consente di avere classi runtime diverse per ogni applicazioni, diverso CLASSPATH per ogni applicazione)
- **web.xml:** file contenente le definizioni delle servlet

Modulo 2
Hands on Servlet

Creare un Servlet di esempio:

- **Si desidera creare una servlet di esempio**
- **La servlet si chiama “Esempio”**
- **Verrà messa nella directory $CATALINA_HOME/webapps**
- **In particolare:**
  - $CATALINA_HOME/webapps/examples/WEB-INF/classes/ conterrà il codice e bytecode Java
  - $CATALINA_HOME/webapps/examples/WEB-INF/ conterrà informazioni aggiuntive per far funzionare la servlet

Servlet Esempio1.java

```
import java.lang.*;
import java.io.*;
import javax.servlet.*;
public class Esempio1 extends GenericServlet {
    public void service(ServletRequest req,
        ServletResponse res){
        try {
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            out.println("Buh!");
        }
        catch (Exception e){ }
    }
}
```

Rendere la Servlet eseguibile

- **Configurare il compilatore**
  - La variabile CLASSPATH deve contenere i file jar delle servlet

    → export CLASSPATH=$CLASSPATH:$CATALINA_HOME/lib/servlet-api.jar
- **Compilare la servlet**
  - javac Esempio1.java

    → si crea Esempio1.class

Rendere la Servlet eseguibile

- **Inserire la servlet nel file web.xml**
  - File: $CATALINA_HOME/webapps/servlets-examples/WEB-INF/web.xml

```
<servlet>
    <servlet-name>Esempio1</servlet-name>
    <servlet-class>Esempio1</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Esempio1</servlet-name>
    <url-pattern>/servlets/servlet/Esempio1</url-pattern>
</servlet-mapping>
```

Uso di HttpServlet

- **Al posto della classe GenericServlet, è possibile utilizzare la classe HttpServlet**
- **Vantaggi:**
  - interfaccia più completa per l'accesso ai parametri della richiesta HTTP
    - supporto per le sessioni (getSession())
    - gestione diretta dei metodi HTTP
      - doGet(), doPost(), doPut(), ...

La servlet Esempio2.java

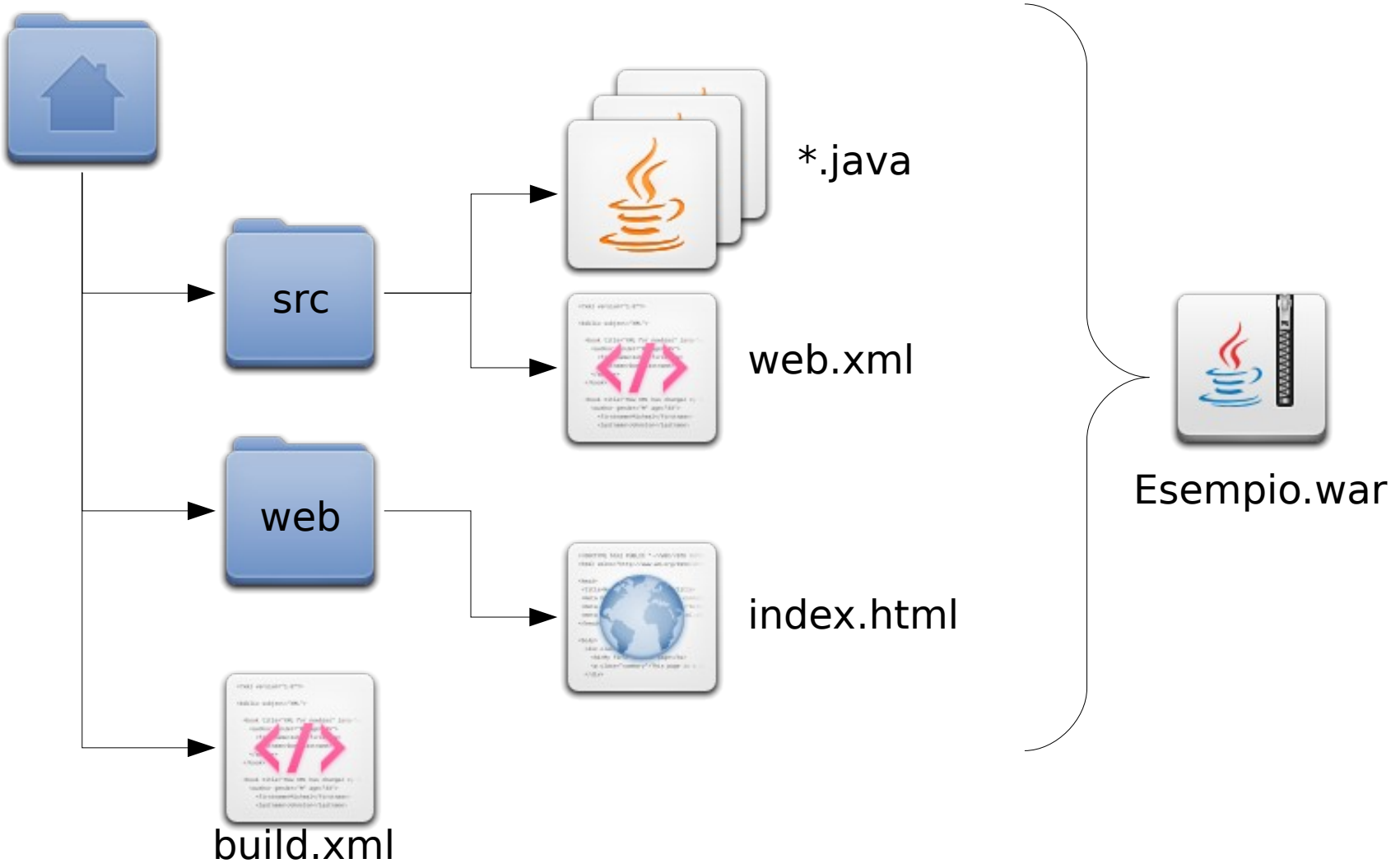
```
import java.lang.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Esempio2 extends HttpServlet {
    public void doGet(HttpServletRequest req,
        HttpServletResponse res){
        try {
            res.setContentType("text/html");
            HttpSession session = req.getSession(false);
            PrintWriter out = res.getWriter();
            out.println("Buh!");
        }
        catch (Exception e){ }
    }
}
```

Automatizzare il build

- **Nell'esercitazione abbiamo fatto molte cose discutibili**
  - Compilazione manuale delle classi
  - Deployment manuale dell'applicazione
- **Soluzione:**
  - Automatizzare il building del progetto
  - Creare un archivio .war di cui è facile il deployment

Preparazione

- **Struttura delle directory:**



Il file index.html

- **Semplice file con i link alle servlet create**

```
<html>
<head>
	<title>Primi esempi di programmazione Servlet</title>
</head>
<body>
	<h1>Primi esempi di programmazione Servlet</h1>
	<ul>
		<li><a href="Esempio1">Esempio con
  GenericServlet</a></li>
		<li><a href="Esempio2">Esempio con HTTPServlet</a></li>
	</ul>
</</body>
</html>
```

Il file web.xml

- **Mapping delle servlet nel server**

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                     http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 version="3.0">
<servlet>
   <servlet-name>Esempio1</servlet-name>
   <servlet-class>Esempio1</servlet-class>
</servlet>
<servlet>
   <servlet-name>Esempio2</servlet-name>
   <servlet-class>Esempio2</servlet-class>
</servlet>
...
```

Il file web.xml

- **Mapping delle servlet nel server**

```
...
<servlet-mapping>
   <servlet-name>Esempio1</servlet-name>
   <url-pattern>/Esempio1</url-pattern>
</servlet-mapping>
<servlet-mapping>
   <servlet-name>Esempio2</servlet-name>
   <url-pattern>/Esempio2</url-pattern>
</servlet-mapping>
</web-app>
```

Automatizzare la compilazione

- **Uso di apache ant**
- **Simile a make, ma orientato a java**
- **Makefile → build.xml**

Il file build.xml

```
<?xml version="1.0"?>

<project name="Esempio" default="all" basedir=".">
    <property name="J2EEdir" value="${basedir}/../tomcat7/lib/"/>
    <property name="dirs.base" value="${basedir}"/>
    <property name="classdir" value="${dirs.base}/build/src"/>
    <property name="src" value="${dirs.base}/src"/>
    <property name="deploymentdescription"
  value="${dirs.base}/src"/>
    <property name="warFile" value="Esempio.war"/>
    <property name="warDir" value="${dirs.base}/build/war"/>
    <property name="web" value="${dirs.base}/web"/>
```

Il file build.xml

```
<!-- Main target  -->
    <target name="all" depends="init,build,buildWar"/>
    <!-- Init -->
    <target name="init">
        <!-- Create web-inf and classes directories -->
        <mkdir dir="${classdir}"/>
        <mkdir dir="${warDir}/WEB-INF"/>
        <mkdir dir="${warDir}/WEB-INF/classes"/>
    </target>
```

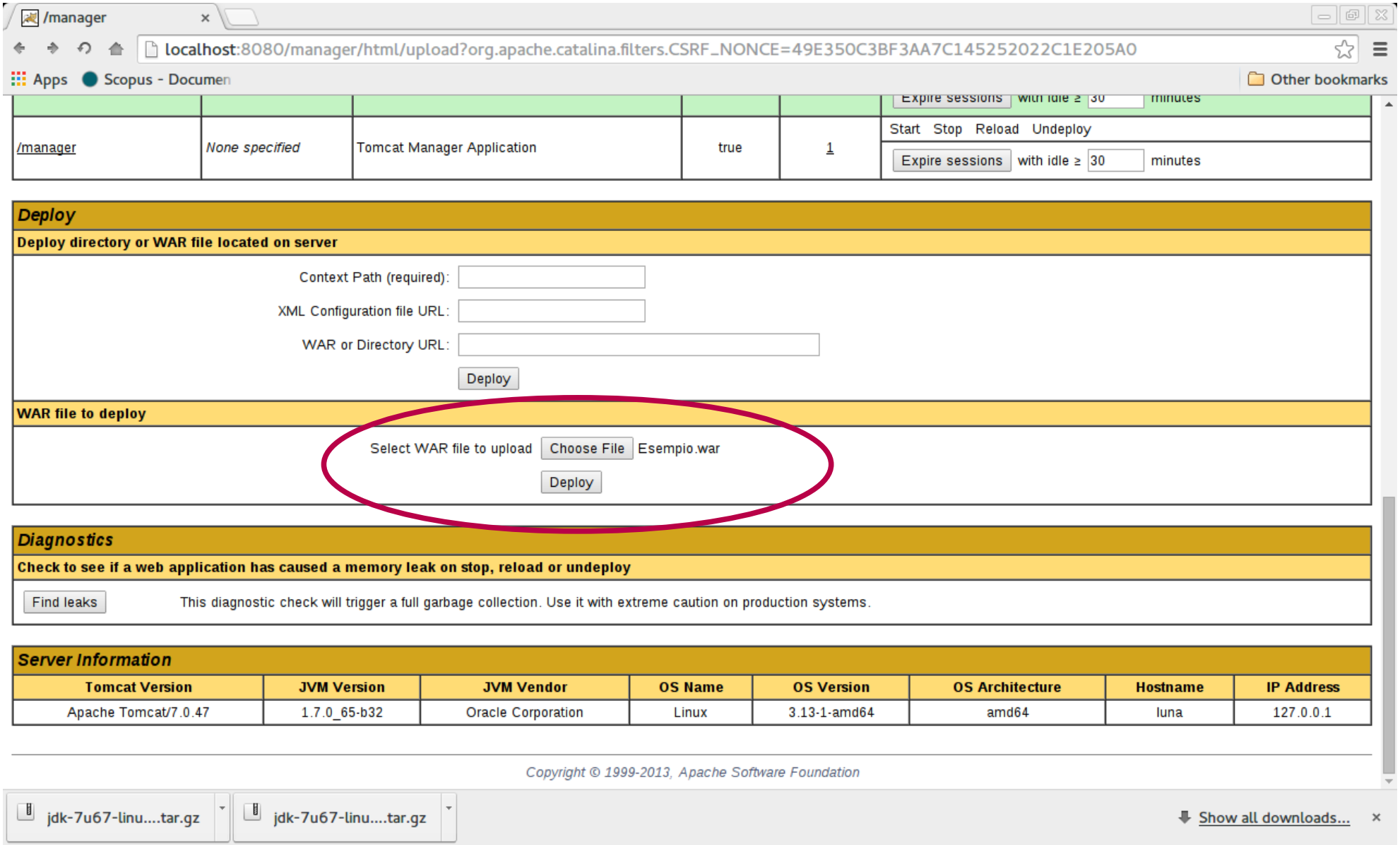
Il file build.xml

```
<!-- Compile Java Files and store in /build/src directory  -->
<target name="build" >
        <javac srcdir="${src}" destdir="${classdir}" debug="true"
includes="**/*.java" >
                <classpath>
                        <pathelement path="${classpath}"/>
                        <fileset dir="${J2EEdir}">
                                <include name="**/servlet-api.jar"/>
                        </fileset>
                </classpath>
        </javac>
</target>
```

Il file build.xml

```
    <!-- Create the web archive File -->
    <target name="buildWar" depends="init">
        <copy todir="${warDir}/WEB-INF/classes">
            <fileset dir="${classdir}" includes="**/*.class" />
        </copy>
        <copy todir="${warDir}/WEB-INF">
            <fileset dir="${deploymentdescription}/"
  includes="web.xml" />
        </copy>
        <copy todir="${warDir}">
            <fileset dir="${web}" includes="**/*.*" />
        </copy>
        <!-- Create war file and place in ear directory -->
        <jar jarfile="${basedir}/${warFile}" basedir="${warDir}" />
    </target>
</project>
```

Deployment


/manager
localhost:8080/manager/html/upload?org.apache.catalina.filters.CSRF_NONCE=49E350C3BF3AA7C145252022C1E205A0
Apps
Scopus - Documen
Other bookmarks
/manager
None specified
Tomcat Manager Application
true
1
Start Stop Reload Undeploy
Expire sessions with idle ≥ 30 minutes
Deploy
Deploy directory or WAR file located on server
Context Path (required):
XML Configuration file URL:
WAR or Directory URL:
Deploy
WAR file to deploy
Select WAR file to upload
Choose File
Esempio.war
Deploy
Diagnostics
Check to see if a web application has caused a memory leak on stop, reload or undeploy
Find leaks
This diagnostic check will trigger a full garbage collection. Use it with extreme caution on production systems.
Server Information
Tomcat Version
JVM Version
JVM Vendor
OS Name
OS Version
OS Architecture
Hostname
IP Address
Apache Tomcat/7.0.47
1.7.0_65-b32
Oracle Corporation
Linux
3.13-1-amd64
amd64
luna
127.0.0.1
Copyright © 1999-2013, Apache Software Foundation
jdk-7u67-linu....tar.gz
jdk-7u67-linu....tar.gz
Show all downloads...