**How to Enable cross domain access in java web-services**

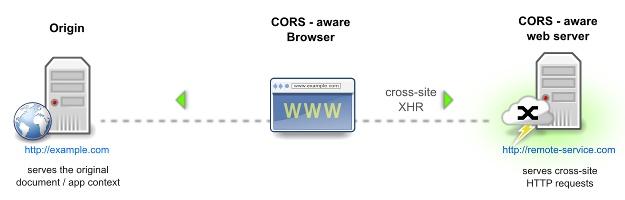
[**http://software.dzhuvinov.com/cors-filter.html**](http://software.dzhuvinov.com/cors-filter.html)

[**http://software.dzhuvinov.com/cors-filter-installation.html**](http://software.dzhuvinov.com/cors-filter-installation.html)

# [**CORS Filter**](http://software.dzhuvinov.com/cors-filter.html)

# The first universal CORS implementation for Java web apps

**CORS Filter is the first universal solution for fitting** [**Cross-Origin Resource Sharing**](http://www.w3.org/TR/cors/) **(CORS) support to Java web applications. CORS is a recent** [**W3C**](http://www.w3.org/) **effort to introduce a standard mechanism for enabling cross-domain requests from web browsers to servers that wish to handle them.**



### The future of the web is cross-domain, not same-origin

**Since the early days of the web (think** [**Netscape 2.0**](http://en.wikipedia.org/wiki/Netscape_Navigator)**) browsers have enforced, to various degrees, a** [**same origin policy**](http://en.wikipedia.org/wiki/Same_origin_policy) **to prevent leaking of confidential user data to third party sites. The same origin policy was carried over to the revolutionary** [**XMLHttpRequest**](http://en.wikipedia.org/wiki/XMLHttpRequest) **which appeared in the early 2000's. Modern web applications, however, increasingly seek to dynamically integrate content and services from third parties, which was initially achieved through "hacks" such as** [**JSONP**](http://en.wikipedia.org/wiki/JSON#JSONP)**. CORS was created in recognition that cross-domain requests advance the spirit of the web, are here to stay, and therefore they'd better be standardised.**

### 

### The philosophy of CORS

**CORS works two-fold:**

* **From a browser script perspective: By allowing cross-domain requests, which are subject to tighter controls on the types of data that is exchanged.** [**Cookies**](http://en.wikipedia.org/wiki/HTTP_cookie)**, for instance, are blocked unless specifically requested by the XHR author *and* allowed by the cross-domain web service. This is done to mitigate the risk of data leaks.**
* **From a web service perspective: By utilising the** [**origin URL**](http://tools.ietf.org/html/draft-abarth-origin) **reported by the browser the target cross-domain web service can determine, based on its origin policy, whether to allow or deny the request.**

**The complete CORS specification is available at** [**http://www.w3.org/TR/cors/**](http://www.w3.org/TR/cors/)

**Note that in order for CORS to work, it must be supported by both browser and web server.**

### Security

**Bear in mind that CORS is not about providing server-side security. The** [**Origin**](http://www.w3.org/TR/cors/#origin-request-header) **request header is produced by the browser and the server has no direct means to verify it.**

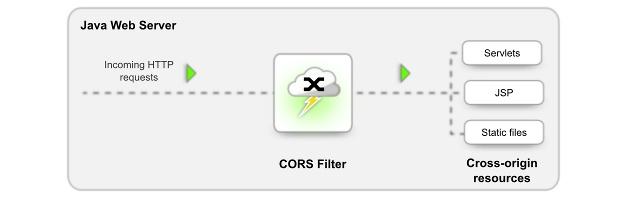
### Browsers supporting CORS

**All major browsers support CORS. The reported penetration among users is at** [**89% as of November 2013**](http://caniuse.com/cors)**.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Firefox | IE | Chrome | Safari | Opera |
| **Firefox 3.5+** | **Internet Explorer 8+✝** | **Google Chrome 3+** | **Apple Safari 4+** | **Opera 12+** |

**✝ Partial support via the** [**XDomainRequest**](http://blogs.msdn.com/b/ieinternals/archive/2010/05/13/xdomainrequest-restrictions-limitations-and-workarounds.aspx) **object. Version 10 of IE is expected to have full CORS support integrated into the common XMLHttpRequest object.**

### The CORS Filter solution - plug in and forget



**The CORS Filter can be plugged into any standard Java Servlet container to handle cross-site requests to servlets, JSPs and HTML files residing on the server.**

**The CORS Filter, as the name implies, implements the clever** [**javax.servlet.Filter**](http://download.oracle.com/javaee/6/api/javax/servlet/Filter.html) **interface. It intercepts incoming HTTP requests and if they are identified as cross-origin, it applies the proper CORS policy and headers, before passing them on to the actual targets (servlets, JSPs, static XML/HTML documents).**

**This transparent nature of the CORS Filter makes it very easy to retrofit existing Java web services with a CORS capability. Just put the CORS JAR file into your CLASSPATH and enable it with a few lines of XML in your web.xml file. The CORS Filter implementation is extremely efficient too - it takes less than 30K of bytecode.**

### 

### CORS Filter documentation

* [**Installation**](http://software.dzhuvinov.com/cors-filter-installation.html)
* [**Configuration**](http://software.dzhuvinov.com/cors-filter-configuration.html)
* [**Request tagging**](http://software.dzhuvinov.com/cors-filter-tagging.html)
* [**Technical specification**](http://software.dzhuvinov.com/cors-filter-spec.html)
* [**Tips and tricks**](http://software.dzhuvinov.com/cors-filter-tips.html)
* [**Browser bugs and quirks**](http://software.dzhuvinov.com/cors-filter-browser-bugs.html)

# [**CORS Filter**](http://software.dzhuvinov.com/cors-filter.html) **Installation**

## Installation

The CORS Filter can run in any [Java Servlet 2.5+](http://en.wikipedia.org/wiki/Java_Servlet) compatible web container, such as the popular open source [Apache Tomcat](http://tomcat.apache.org/) server. Installation is a straightforward 3-step process.

### 1. Place the CORS JAR and its dependency in the CLASSPATH

[Download](http://software.dzhuvinov.com/download.html#download-cors) the cors-filter-<version>.jar file and its java-property-utils-<version>.jardependency, and put them into the [CLASSPATH](http://en.wikipedia.org/wiki/Classpath_(Java)) of your web server.

[cors-filter-1.8.jar](http://search.maven.org/remotecontent?filepath=com/thetransactioncompany/cors-filter/1.8/cors-filter-1.8.jar) [java-property-utils-1.9.jar](http://search.maven.org/remotecontent?filepath=com/thetransactioncompany/java-property-utils/1.9/java-property-utils-1.9.jar)  
If you have Apache Tomcat there are two CLASSPATH choices: If you intend to use CORS with a single web application put the JAR file in

$CATALINA\_HOME/webapps/<your-web-app>/WEB-INF/lib/  
To make CORS available globally, to all web applications, place the JAR in $CATALINA\_HOME/lib/  
**2. Add CORS configuration to web.xml**

Open the WEB-INF/web.xml file of the web application where you intend to enable CORS and add a CORS Filter [declaration](http://download.oracle.com/docs/cd/E13222_01/wls/docs81/webapp/web_xml.html#1015950) and [mapping](http://download.oracle.com/docs/cd/E13222_01/wls/docs81/webapp/web_xml.html#1039330).

The XML declaration to load the CORS filter:

<filter>  
 <filter-name>CORS</filter-name>  
 <filter-class>com.thetransactioncompany.cors.CORSFilter</filter-class>  
</filter>

Then declare a filter mapping to tell the web server which servlets or URLs should be cross-domain-request enabled.

Example of applying the CORS filter to a single servlet:

<filter-mapping>  
 <filter-name>CORS</filter-name>  
 <servlet-name>MyServlet</servlet-name>  
</filter-mapping>

And how to apply the CORS filter to all web app URLs:

<filter-mapping>  
 <filter-name>CORS</filter-name>  
 <url-pattern>/\*</url-pattern>  
</filter-mapping>  
Have a look at the [web.xml](http://software.dzhuvinov.com/files/cors/web.xml.txt) of the demo CORS application included with the download package to see a complete CORS filter declaration and mapping example.

Finally, remember to restart your web server for the installation to take effect.

**Important note:** By default the CORS Filter will apply a "public access" CORS policy, allowing all cross-site requests through (including credentials/cookies). Leaving the CORS Filter at this setting would actually be fine for most situations as CORS is not about adding server security; its primary intent is to protect the browser - the legitimate JavaScript apps running in it and the user's confidential data, such as cookies.

If you want to modify the default CORS Filter behaviour, proceed to the [configuration instructions](http://software.dzhuvinov.com/cors-filter-configuration.html).

### Resources

**Official W3C documents:**

* [**The W3C Working Draft on Cross-Origin Resource Sharing**](http://www.w3.org/TR/cors/)

**Useful notes, tips and tricks:**

* [**How to detect CORS support in a browser**](http://blog.dzhuvinov.com/?p=848)
* [**How to debug CORS requests with FireBug**](http://blog.dzhuvinov.com/?p=685)
* [**Mozilla notes on cross-domain XHR**](https://developer.mozilla.org/En/XMLHttpRequest/Using_XMLHttpRequest#section_16)