# Cross Origin Resource Sharing (CORS)

[Cross-Origin Resource Sharing](http://www.w3.org/TR/access-control/) (CORS) is a W3C Working Draft that defines how the browser and server must communicate when accessing sources across origins.

## Use Case

The site abc.appspot.com has some data that the site xyz.appspot.com wants to access. This type of request traditionally wouldn’t be allowed under the browser’s [same origin policy](http://en.wikipedia.org/wiki/Same_origin_policy). By supporting CORS requests, abc.appspot.com can add a few special response headers that allow xyz.appspot.com to access the data.

## Making a CORS Request

CORS is supported in the following browsers:

* Chrome 3+
* Firefox 3.5+
* Opera 12+
* Safari 4+
* Internet Explorer 8+

Chrome, Firefox, Opera and Safari all use the [XMLHttpRequest2 object](http://www.html5rocks.com/en/tutorials/file/xhr2/). Internet Explorer uses the similar XDomainRequest object.

## Sample JQUERY code to generate CORS request

$(document).ready(**function**(){

//GET CORS request

$("#jsonbttn").click(**function**(){

$.ajax( {

type: "GET",

crossDomain:**true**,

url: "http://arindam-source1.appspot.com/api/client/v02/magazine/list",

success: **function** (data) {

console.log(data);

},

error: **function**( xhr, textStatus, errorThrown ) {

console.log(xhr);

console.log( "HTTP Status: " + xhr.status );

console.log( "Error textStatus: " + textStatus );

console.log( "Error thrown: " + errorThrown );

}

});

});

//POST CORS request

$("#btn").click(**function**(){

**var** vname = $("#username").val();

**var** vpass = $("#password").val();

**if**(vname=='' && vpass=='')

{

alert("Please fill out the form");

}

**else** **if**(vname=='' && vpass!==''){alert('Name field is required')}

**else** **if**(vpass=='' && vname!==''){alert('Password field is required')}

**else**{

$.ajax( {

cache: **false**,

crossDomain: **true**,

dataType: "json",

url: "http://arindam-source1.appspot.com/api/client/v02/authentication/login",

data: {"username": vname, "password": vpass},

type: "POST",

success: **function**( jsonObj, textStatus, xhr ) {

console.log(jsonObj);

},

error: **function**( xhr, textStatus, errorThrown ) {

console.log( "HTTP Status: " + xhr.status );

console.log( "Error textStatus: " + textStatus );

console.log( "Error thrown: " + errorThrown );

}

} );

}

});

});

## Generate response of a simple request in the server using Jersey

The following code sample adds a doOptions method within a servlet and sends the appropriate headers in response to a [CORS preflight request](http://www.w3.org/TR/cors/#resource-preflight-requests).

**import** javax.ws.rs.container.ContainerRequestContext;

**import** javax.ws.rs.container.ContainerResponseContext;

**import** javax.ws.rs.container.ContainerResponseFilter;

**import** javax.ws.rs.container.PreMatching;

**import** javax.ws.rs.ext.Provider;

@Provider

@PreMatching

**public** **class** ResponseCorsFilter **implements** ContainerResponseFilter {

@Override

**public** **void** filter(ContainerRequestContext requestCtx,

ContainerResponseContext responseCtx) {

responseCtx.getHeaders().add("Access-Control-Allow-Origin", "\*");

responseCtx.getHeaders().add("Access-Control-Allow-Methods",

"GET, POST, DELETE, PUT");

}

}

Access-Control-Allow-Origin: \*

Access-Control-Allow-Credentials: true

Access-Control-Allow-Methods: GET, POST, DELETE, PUT

These sets of additional HTTP Headers that could be included as part of the HTTP response when it goes back to the browser by having it included in a class which implements ContainerResponseFilter.

Standard CORS requests do not send or set any cookies by default. In order to include cookies as part of the request, you need to set the **XMLHttpRequest’s.withCredentials** property to true

In order for this to work, the server must also enable credentials by setting the **Access-Control-Allow-Credentials** response header to “true”.

**Dealing with Browser Preflight Request HTTP Method: OPTIONS**

Following code deals with the browser preflight requests.

**import** javax.ws.rs.container.ContainerRequestContext;

**import** javax.ws.rs.container.ContainerRequestFilter;

**import** javax.ws.rs.container.PreMatching;

**import** javax.ws.rs.core.Response;

**import** javax.ws.rs.ext.Provider;

@Provider

@PreMatching

**public** **class** RequestCorsFilter **implements** ContainerRequestFilter {

@Override

**public** **void** filter(ContainerRequestContext requestCtx) **throws** IOException {

// When HttpMethod comes as OPTIONS, just acknowledge that it accepts...

**if** (requestCtx.getRequest().getMethod().equals("OPTIONS")) {

// Just send a OK signal back to the browser

requestCtx.abortWith(Response.*status*(Response.Status.*OK*).build());

}

}

}