



softcontext@gmail.com

Ajax

Ajax(**A**synchronous **J**avaScript and **X**ML)는 대화식 웹 애플리케이션의 제작을 위해 아래와 같은 조합을 이용하는 웹 개발 기법이다.

참고: <https://ko.wikipedia.org/wiki/Ajax>

- 표현 정보를 위한 HTML (또는 XHTML) 과 CSS
- 동적인 화면 출력 및 표시 정보와의 상호작용을 위한 DOM, 자바스크립트
- 웹 서버와 비동기적으로 데이터를 교환하고 조작하기 위한 XML, XSLT, XMLHttpRequest (Ajax 애플리케이션은 XML/XSLT 대신 미리 정의된 HTML 이나 일반 텍스트, JSON, JSON-RPC 를 이용할 수 있다).

DHTML 이나 LAMP 와 같이 Ajax 는 자체가 하나의 특정한 기술을 말하는 것이 아니며, 함께 사용하는 기술의 묶음을 지칭하는 용어이다. 실제로 AFLAX 와 같이 사실상 Ajax 에 바탕을 두고 있는 유사/복합 기술들이 속속 나타나고 있다.

Ajax 애플리케이션은 실행을 위한 플랫폼으로 위에서 열거한 기술들을 지원하는 웹 브라우저를 이용한다. 이것을 지원하는 브라우저로는 모질라 파이어폭스, 인터넷 익스플로러, 오페라, 사파리, 구글 크롬 등이 있다. 단, 오페라는 현재 XSL 포매팅 객체와 XSLT 변환을 지원하지 않는다.

기존 기술과의 차이점

기존의 웹 애플리케이션은 브라우저에서 폼을 채우고 이를 웹 서버로 제출(submit)을 하면 하나의 요청으로 웹 서버는 요청된 내용에 따라서 데이터를 가공하여 새로운 웹 페이지를 작성하고 응답으로 되돌려준다. 이때 최초에 폼을 가지고 있던 페이지와 사용자가 이 폼을 채워 결과물로서 되돌려 받은 페이지는 일반적으로 유사한 내용을 가지고 있는 경우가 많다. 결과적으로 중복되는 HTML 코드를 다시 한번 전송을 받음으로써 많은 대역폭을 낭비하게 된다. 대역폭의 낭비는 금전적 손실을 야기할 수 있으며 사용자와 대화(상호 반응)하는 서비스를 만들기 어렵게도 한다.

반면에 Ajax 애플리케이션은 필요한 데이터만을 웹서버에 요청해서 받은 후 클라이언트에서 데이터에 대한 처리를 할 수 있다. 보통 SOAP 이나 XML 기반의 웹 서비스 프로토콜이 사용되며, 웹 서버의 응답을 처리하기 위해 클라이언트 쪽에서는 자바스크립트를 쓴다. 웹 서버에서 전적으로

처리되던 데이터 처리의 일부분이 클라이언트 쪽에서 처리 되므로 웹 브라우저와 웹 서버 사이에 교환되는 데이터량과 웹서버의 데이터 처리량도 줄어들기 때문에 애플리케이션의 응답성이 좋아진다. 또한 웹서버의 데이터 처리에 대한 부하를 줄여주는 일이 요청을 주는 수많은 컴퓨터에 대해서 일어나기 때문에 전체적인 웹 서버 처리량도 줄어들게 된다.

장점

- 페이지 이동없이 고속으로 화면을 전환할 수 있다.
- 서버 처리를 기다리지 않고, 비동기 요청이 가능하다.
- 수신하는 데이터 양을 줄일 수 있고, 클라이언트에게 처리를 위임할 수도 있다.

단점

- Ajax 를 쓸 수 없는 브라우저에 대한 문제가 있다.
- Http 클라이언트의 기능이 한정되어 있다.
- 페이지 이동없는 통신으로 인한 보안상의 문제
- 지원하는 Charset 이 한정되어 있다.
- 스크립트로 작성되므로 Debugging 이 용이하지 않다.
- 요청을 남발하면 역으로 서버 부하가 늘 수 있음.
- 동일 출처 정책(Same Origin Policy)으로 인해 다른 도메인과는 통신이 불가능하다.

참고: https://developer.mozilla.org/ko/docs/Web/Security/Same-origin_policy

Origin Determination Rules

http://www.example.com/dir/page.html 문자열을 URL 로 사용하여 서버로부터 page.html 을 받아 온 경우 동일출처 정책은 다음과 같이 판단한다.

참고: https://en.wikipedia.org/wiki/Same-origin_policy

Compared URL	Outcome	Reason
http://www.example.com/dir/page2.html	Success	Same protocol, host and port
http://www.example.com/dir2/other.html	Success	Same protocol, host and port
http://username:password@ www.example.com/dir2/other.html	Success	Same protocol, host and port
http://www.example.com:81/dir/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir/other.html	Failure	Different protocol
http://en.example.com/dir/other.html	Failure	Different host
http://example.com/dir/other.html	Failure	Different host (exact match required)
http://v2.www.example.com/dir/other.html	Failure	Different host (exact match required)
http://www.example.com:80/dir/other.html	Depends	Port explicit. Depends on implementation in browser.

Ajax 의 보급

이것은 이미 존재하던 기술이었지만, 2005 년 초에 있었던 몇 가지 사례 이후로 인기를 끌기 시작했다. 먼저 구글이 구글 그룹스를 포함한 훌륭한 대화형 애플리케이션의 기반을 위해 비동기식 통신을 이용한 것이다. 두 번째로는 Ajax 라는 용어가 AJAX: A new approach for a new application 기사에서 등장한 것으로, 이후 빠르게 대중화되어 이 기법의 보급에 도움이 되었다.

현재 대화형 웹 페이지를 위한 도구로서 Ajax 를 이용하는 애플리케이션들이 급격히 늘어나고 있으며, 이는 부분적으로 이용할 수 있는 애플리케이션 툴킷(예: Ruby on Rails, DWR)이 늘어나 프로그래머들이 구현하기가 쉬워진 때문이다.

대한민국에서도 네이버와 다음등을 비롯한 포털 업체에서 이 기술을 도입하고 있다.

Ajax 와 접근성

Ajax 사용에 있어서, Ajax 기술을 지원하지 않는 브라우저를 위한 대체물을 만드는 것은 거의 힘들다. 이 같은 한계는 WAI 접근성 지침에 거스르는 측면이 있다.

한편, 웹 개발자들은 때때로 Ajax 를 단순히 웹 페이지의 일부분을 대체하기 위해 사용한다. 비 AJAX 사용자가 전체 페이지를 불러오는 것에 비해 Ajax 사용자는 페이지의 일부분만을 불러올 수가 있다. 이것으로 개발자들이 비 AJAX 환경에 있는 사용자의 접근성을 포함한 경험을 보호할 수 있으며, 적절한 브라우저를 이용하는 경우에 전체 페이지를 불러오는 일 없이 응답성을 향상시킬 수 있다.

웹 브라우저

Ajax 를 지원하는 웹 브라우저

참고: 이 목록은 포괄적이며, 브라우저 특징에 따라서 Ajax 애플리케이션이 지원이 다를 수 있다.

- 마이크로소프트 인터넷 익스플로러 5.0 이상, 기타 인터넷 익스플로러를 기반으로 한 브라우저들 (맥 OS 버전은 미지원)
- 게코 기반 모질라, 모질라 파이어폭스, 시몽키, 에피파니, 갈레온 브라우저 그리고 넷스케이프 7.1 이상
- KHTML 3.2 이상, Konqueror 3.2 이상 그리고 애플 사파리 1.2 이상 포함
- 오페라 브라우저 8.0 이상, 오페라 모바일 브라우저 8.0 이상
- 구글 크롬

Ajax 를 지원하지 않는 웹 브라우저

참고: 이 목록의 브라우저는 확실히 Ajax 를 지원하지 않는다.

- 오페라 7 이하
- 마이크로소프트 인터넷 익스플로러 4.0 이하
- 텍스트 기반의 브라우저 링크스, w3m

- 시각장애인을 위한 브라우저
- 1997 년 이전 브라우저

비동기 프로그래밍

브라우저 환경내에 자바스크립트는 쓰레딩 모델에 접근할 수 없다. DOM 객체를 다루는 모든 작업은 싱글쓰레드 기반으로 돌아간다. 브라우저는 일반적으로 사용되는 XHR(XMLHttpRequest 또는 AJAX) API 와 같은 몇 가지 비동기 API 들을 제공한다. 몇 가지 예를 들면 IndexedDB, SQLite, HTML5 Web workers, 그리고 HTML5 GeoLocation API 가 있다.

브라우저 환경에서 처리되는 자바스크립트 코드에서 비동기 프로그래밍 로직을 사용하는 방법은 이벤트 또는 콜백을 사용하는 것이다. 이벤트 기반 비동기 API 의 경우, 개발자가 특정 DOM 객체에 이벤트 핸들러를 등록하여 이벤트 발생 시 원하는 동작을 처리한다. 브라우저는 보통 다른 쓰레드로 그 동작을 수행할 것이고 적절한 때에 메인 쓰레드에서 이벤트를 발생시킬 것이다.

비동기 프로그래밍의 필요성

src/main/webapp/public/html/example-webworker-needs.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script src="http://code.jquery.com/jquery-3.1.1.min.js"
    integrity="sha256-hVVnYaiADRTO2PzUGmuLJR8BLUSjGIZsDYGmJLv2b8="
    crossorigin="anonymous"> </script>
<script type="text/javascript">
    $(function(){
        $('#calculate').click(function(){
            setTimeout(function() {
                var sum = 0;
                for(var i=1; i <= 2000000000; i++){
                    sum += i;
                }
            }, 1000);
        });
    });
</script>
```

```

        $('#target').text('result = '+sum);
    }, 1);
});

$('#alive').click(function(){
    alert('yes, i am alive');
});
});
</script>
</head>
<body>

    <button id="calculate">get sum of 1 to 2,000,000,000</button>
    <button id="alive">a u alive?</button>

    <hr>

    <div id="target"> </div>

</body>
</html>

```

"get sum of 1 to 2,000,000,000" 버튼을 누르면 결과가 표시되기 위해 5 초 이상이 필요하다. 그 사이에 "a u alive?" 버튼을 누르면 아무리 반응을 얻을 수 없다. 브라우저 환경내에서 자바스크립트는 싱글스레드만 존재하기 때문이다. 긴 시간이 소요되는 로직은 사용자의 요청에 바로 응답하지 못하는 현상을 일으키기 때문에 각별한 주의가 필요하다. 어쩔 수 없이 오랜 시간이 걸리는 로직을 처리해야 한다면 몇가지 대안을 생각해 볼 수 있다.

- 오랜 시간이 걸리는 작업을 분할해서 수행하고 각 결과를 합쳐서 처리한다.
- 최신 브라우저가 지원하는 Web Worker 를 사용한다.

Web Workers

A web worker is a JavaScript running in the background, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

웹워커 지원브라우저 확인

http://www.w3schools.com/html/html5_webworkers.asp

src/main/webapp/public/html/example-worker.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script src="http://code.jquery.com/jquery-3.1.1.min.js"
    integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmJLv2b8="
    crossorigin="anonymous"> </script>
<script type="text/javascript">

    $(function(){
        $('#start').click(function(){
            startWorker();
        });

        $('#stop').click(function(){
            stopWorker();
        });

        $('#alive').click(function(){
            alert('yes, i am alive');
        });
    });
```

```

var w;

function startWorker() {
    if(typeof(Worker) !== "undefined") {
        if(typeof(w) === "undefined") {
            w = new Worker("../js/example-worker.js");
        }
        w.onmessage = function(event) {
            $('#target').text('result = '+event.data);
        };
    } else {
        $('#target').text('Sorry, your browser does not support Web Worker');
    }
}

function stopWorker() {
    w.terminate();
    w = undefined;
}
</script>
</head>
<body>

    <h2>get sum of 1 to 2,000,000,000 </h2>
    <button id="start">start</button>
    <button id="stop">stop</button>
    <button id="alive">alive?</button>
    <hr>
    <div id="target"></div>

</body>
</html>

```

src/main/webapp/public/js/example-worker.js

```
function delegate(max){  
  setTimeout(function() {  
    var sum = 0;  
    for (var i = 1; i <= max; i++) {  
      sum += i;  
    }  
    postMessage(sum);  
  }, 500);  
}  
  
delegate(2000000000);
```

첫 예제

아래는 GET 을 사용하여 Ajax 요청을 하는 단순한 예제이다.

STS > Spring Boot Project > 디펜던시 web 선택

프로젝트명: javascript-ajax

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>javascript-ajax</name>
  <description></description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.4.1.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
```

```
<java.version>1.8</java.version>
</properties>

<dependencies>

    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>

    <!-- spring-boot-starter-web 에 포함된 tomcat 은 JSP 엔진이 없다. -->
    <dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-jasper</artifactId>
        <scope>provided</scope>
    </dependency>

    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
    </dependency>
```

```
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>

</project>
```

application.properties

```
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
```

DefaultViewConfigurer.java

```
package com.example.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.core.Ordered;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@Configuration
public class DefaultViewConfigurer extends WebMvcConfigurerAdapter {

    @Override
    public void addViewControllers(ViewControllerRegistry registry) {
```

```
        registry.addViewController("/").setViewName("forward:/public/html/example-client.html");
        registry.setOrder(Ordered.HIGHEST_PRECEDENCE);
        super.addViewControllers(registry);
    }
}
```

ExampleController.java

```
package com.example.common.controller;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class ExampleController {

    private Logger logger = LoggerFactory.getLogger(this.getClass());

    @RequestMapping(value={"send-ajax-data"}, method=RequestMethod.GET)
    public String doGet(){
        logger.info("ExampleController.doGet() called");

        return "send-ajax-data";
    }
}
```

src/main/webapp/public/html/example-client.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script type="text/javascript" src="/public/js/example-client.js"></script>
</head>
<body>

    <h1>example-client.html</h1>

</body>
</html>
```

src/main/webapp/public/js/example-client.js

```
// This is the client-side script

// Initialize the Ajax request
var xhr = new XMLHttpRequest();
xhr.open('get', 'send-ajax-data');

// Track the state changes of the request
xhr.onreadystatechange = function() {
    // Ready state 4 means the request is done
    if (xhr.readyState === 4) {
        // 200 is a successful return
        if (xhr.status === 200) {
            // 'This is the returned text.'
            alert(xhr.responseText);
        } else {
```



```
        // An error occurred during the request
        alert('Error: ' + xhr.status);
        console.log(xhr.readyState);
        console.log(xhr.status);
        console.log(xhr.responseText);
    }
}
};

// Send the request to send-ajax-data.jsp
xhr.send(null);
```

src/main/webapp/WEB-INF/jsp/send-ajax-data.jsp

```
<%@ page language="java" contentType="text/plain; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
// This is the server-side script

// Set the content type
response.setContentType("Content-Type: text/plain");

// Send the data back
response.getWriter().append("This is the returned text.");
%>
1234567890
```

테스트



example-client.html

localhost:8080 내용:×

This is the returned text.

1234567890

☐ 이 페이지가 추가적인 대화를 생성하지 않도록 차단합니다.

확인

jQuery 사용 예제

이번 예에서는 jQuery 의 함수 \$.get 을 사용하여 서버와 AJAX 통신을 해보자.

ExampleController.java

```
package com.example.common.controller;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class ExampleController {

    private Logger logger = LoggerFactory.getLogger(this.getClass());

    @RequestMapping(value={"send-ajax-data"}, method=RequestMethod.GET)
    public String doGet(){
        logger.info("ExampleController.doGet() called");

        return "send-ajax-data";
    }

    @RequestMapping(value={"example2"}, method=RequestMethod.GET)
```

```

    public void example2(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{
        logger.info("ExampleController.example2() called");

        String target = "/public/html/example-client2.html";
        request.getRequestDispatcher(target).forward(request, response);
    }
}

```

src/main/webapp/public/html/example-client2.html

```

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"> </script>
<script type="text/javascript" src="/public/js/example-client2.js"> </script>
<script type="text/javascript">
    $(document).ready(function() {
        $("#msgid").html("get message from Server");
        $("#msgid").click(talk);
    });
</script>

</head>
<body>

    <h1>example-client2.html</h1>
    <hr>

    <button id="msgid"> </button>

```

```
</body>
</html>
```

src/main/webapp/public/js/example-client2.js

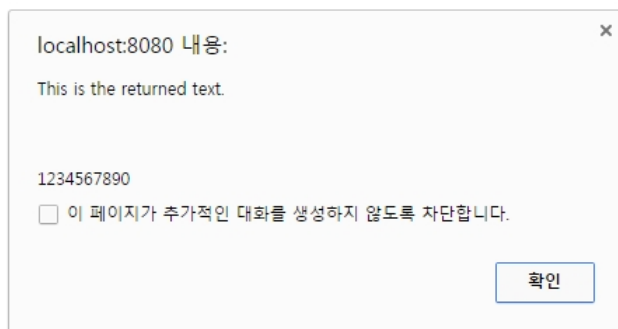
```
function talk() {
    $.get('send-ajax-data', function(data) {
        alert(data);
    });
}
```

테스트

← → ↺ ⬆ localhost:8080/example2

example-client2.html

get message from Server



브라우저가 Ajax 를 지원하는지 체크

STS > Spring Boot > 디펜던시 web 선택

프로젝트명: javascript-ajax-2

StaticResourceConfigurer.java

```
package com.example.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.core.Ordered;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@Configuration
public class StaticResourceConfigurer extends WebMvcConfigurerAdapter {

    @Override
    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController("/").setViewName("forward:/public/html/hello.html");
        registry.setOrder(Ordered.HIGHEST_PRECEDENCE);
        super.addViewControllers(registry);
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/public/*").addResourceLocations("/public/*");
        super.addResourceHandlers(registry);
    }
}
```

src/main/webapp/public/html/hello.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>

    <h1>hello.html</h1>

</body>
</html>
```

정적리소스 접근 테스트



← → ↻ ⬆ localhost:8080

hello.html



← → ↻ ⬆ localhost:8080/public/html/hello.html

hello.html

src/main/webapp/public/html/ajax-able.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>

    <h1>ajax-able.html</h1>
    <script type="text/javascript">
        function isAjaxAvailable() {
            var ajaxRequest;

            try {
                // Opera 8.0+, Firefox, Safari
                ajaxRequest = new XMLHttpRequest();
                return true;
            } catch (e) {
                // Internet Explorer Browsers
                try {
                    ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");
                    return true;
                } catch (e) {
                    try {
                        ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");
                        return true;
                    } catch (e) {
                        // Something went wrong
                        alert("Your browser broke!");
                        return false;
                    }
                }
            }
        }
    </script>

```



```
        }
    }
}
</script>

<div>
    <button onclick="javascript:alert(isAjaxAvailable());">is Ajax available?</button>
</div>

<div>
    <h2>Steps of AJAX Operation</h2>
    1. A client event occurs.<br>
    2. An XMLHttpRequest object is created.<br>
    3. The XMLHttpRequest object is configured.<br>
    4. The XMLHttpRequest object makes an asynchronous request to the Webserver.<br>
    5. The Webserver returns the result containing XML document.<br>
    6. The XMLHttpRequest object calls the callback() function and processes the result.<br>
    7. The HTML DOM is updated.
</div>

</body>
</html>
```

XMLHttpRequest Methods

메소드	기능
abort()	Cancels the current request.
getAllResponseHeaders()	Returns the complete set of HTTP headers as a string.
getResponseHeader(headerName)	Returns the value of the specified HTTP header.
open(method, URL) open(method, URL, async) open(method, URL, async, userName) open(method, URL, async, userName, password)	<p>Specifies the method, URL, and other optional attributes of a request.</p> <p>The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods, such as "PUT" and "DELETE" (primarily used in REST applications) may be possible.</p> <p>The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.</p>
send(content)	Sends the request.
setRequestHeader(label, value)	Adds a label/value pair to the HTTP header to be sent.

XMLHttpRequest Properties

속성	설명												
onreadystatechange	An event handler for an event that fires at every state change.												
readyState	<p>The readyState property defines the current state of the XMLHttpRequest object.</p> <table> <tr> <th>State</th><th>Description</th></tr> <tr> <td>0</td><td>The request is not initialized.</td></tr> <tr> <td>1</td><td>The request has been set up.</td></tr> <tr> <td>2</td><td>The request has been sent.</td></tr> <tr> <td>3</td><td>The request is in process.</td></tr> <tr> <td>4</td><td>The request is completed.</td></tr> </table>	State	Description	0	The request is not initialized.	1	The request has been set up.	2	The request has been sent.	3	The request is in process.	4	The request is completed.
State	Description												
0	The request is not initialized.												
1	The request has been set up.												
2	The request has been sent.												
3	The request is in process.												
4	The request is completed.												
responseText	Returns the response as a string.												
responseXML	Returns the response as XML. This property returns an XML document object, which can be examined and parsed using the W3C DOM node tree methods and properties.												
status	Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").												
statusText	Returns the status as a string (e.g., "Not Found" or "OK").												

Example 01

앞서 만든 프로젝트를 계속 사용하면서 학습한다.

pom.xml

다음 설정을 추가한다.

```
<dependency>
  <groupId>org.apache.tomcat.embed</groupId>
  <artifactId>tomcat-embed-jasper</artifactId>
  <scope>provided</scope>
</dependency>

<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
</dependency>
```

src/main/webapp/public/js/http-request.js

```
function getXMLHttpRequest() {
  if (window.ActiveXObject) { // IE
    try {
      return new ActiveXObject("Msxml2.XMLHTTP");
    } catch (e) {
      try {
        return new ActiveXObject("Microsoft.XMLHTTP"); // IE5
      } catch (e1) {
        return null;
      }
    }
  } else if (window.XMLHttpRequest) { // Others
    return new XMLHttpRequest();
  }
}
```

```

    } else {
        return null;
    }
}

var httpRequest = null;

function sendRequest(url, params, callback, method) {
    httpRequest = getXMLHttpRequest();
    var httpMethod = method ? method : 'GET';
    if (httpMethod != 'GET' && httpMethod != 'POST') {
        httpMethod = 'GET';
    }
    var httpParams = (params == null || params == '') ? null : params;
    var httpUrl = url;
    if (httpMethod == 'GET' && httpParams != null) {
        httpUrl = httpUrl + "?" + httpParams;
    }
    httpRequest.open(httpMethod, httpUrl, true);
    httpRequest.setRequestHeader('Content-Type',
        'application/x-www-form-urlencoded');
    httpRequest.onreadystatechange = callback;
    httpRequest.send(httpMethod == 'POST' ? httpParams : null);
}

```

src/main/webapp/public/example/example-01.html

```

<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
<script type="text/javascript" src="/public/js/http-request.js"> </script>

```

```

<script type="text/javascript">
    function loadNews() {
        // url, params, callback, method
        sendRequest("example-01.jsp", null, loadCallBack, "GET");
    }

    function loadCallBack() {
        if (httpRequest.readyState == 4 && httpRequest.status == 200) {
            document.getElementById("newsList").innerHTML = httpRequest.responseText;
        } else {
            $('#newsList').html('Loading is failed');
        }
    }
}
</script>
</head>
<body>

    <input type="button" value="뉴스 확인" onclick="loadNews()">
    <div id="newsList"></div>

</body>
</html>

```

src/main/webapp/public/example/example-01.jsp

```

<%@ page language="java" contentType="text/plain; charset=utf-8"%>
<%
    String[] news = { "aaa", "bbb", "ccc" };
    int cnt = 1;

    for (String title : news) {
        out.print(cnt + " : ");
        out.print(title + "<br>");
    }
}

```

```
        cnt++;  
    }  
%>
```

테스트



Example 02

src/main/webapp/public/example/example-02.html

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script type="text/javascript">
    $(function() {
        $("button").click(function() {
            $("#target").load("/public/text/example-02.txt");
        });
    });
</script>
</head>
<body>

    <button>Get External Content</button>
    <div id="target"></div>

</body>
</html>
```

src/main/webapp/public/text/example-02.txt

```
<div>
    <h2>What is AJAX?</h2>
    <hr>
```


<p id="p1">AJAX = Asynchronous JavaScript and XML.</p>

In short, AJAX is about loading data in the background and

display it on the webpage, without reloading the whole page.
</div>

테스트



What is AJAX?

AJAX = Asynchronous JavaScript and XML.

In short, AJAX is about loading data in the background and display it on the webpage, without reloading the whole page.

Example 03

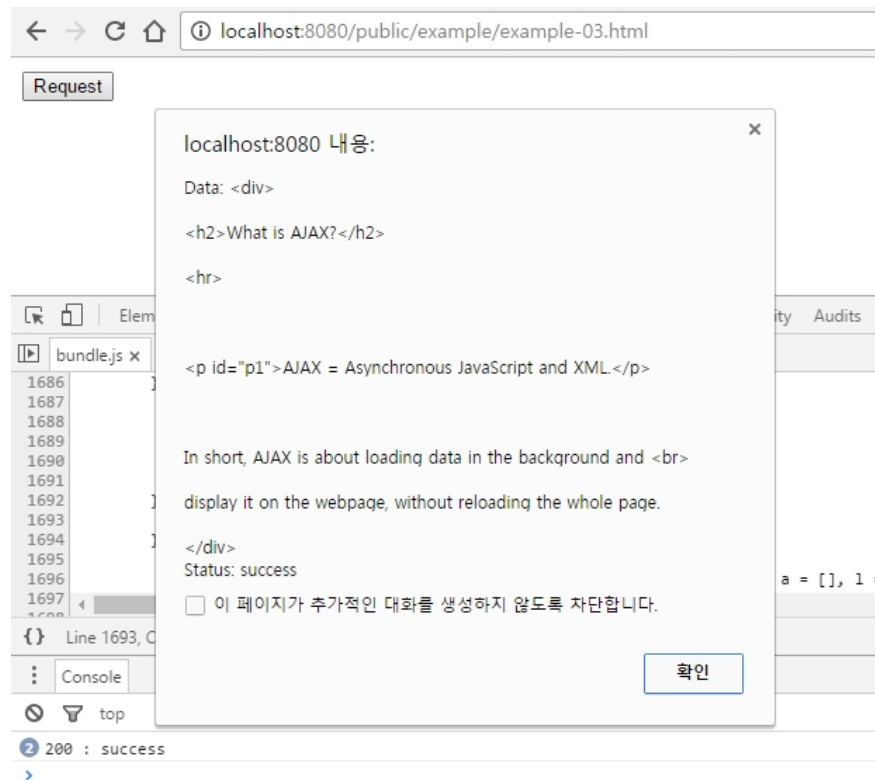
src/main/webapp/public/example/example-03.html

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script type="text/javascript">
    $(function() {
        $("button").click(function() {
            $.get("/public/text/example-02.txt", function(data, status, xhr) {
                alert("Data: " + data + "\nStatus: " + status);
                console.log(xhr.status + " : " + xhr.statusText);
            });
        });
    });
</script>
</head>
<body>

    <button>Request</button>

</body>
</html>
```

테스트



Example 04

src/main/webapp/public/example/example-04.html

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script type="text/javascript">
    $(function() {
        $("button").click(function() {
            $.post("example-04.jsp", {
                name : "HONG Gildong",
                city : "Seoul"
            }, function(data, status, xhr) {
                $("div").html("Data: " + data + "<br>Status: " + status);
                console.log(xhr.status + " : " + xhr.statusText);
            });
        });
    });
</script>
</head>
<body>

    <button>Request</button>
    <div></div>

</body>
</html>
```

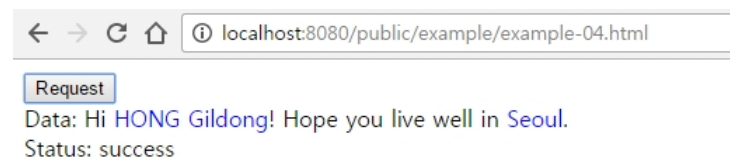
src/main/webapp/public/example/example-04.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"%>
<%
    String name = "";
    String city = "";

    name = request.getParameter("name");
    city = request.getParameter("city");

    out.append("Hi <label style='color:blue;'>" + name + "</label>! ");
    out.append("Hope you live well in <label style='color:blue;'>" + city + "</label>.");
%>
```

테스트



Example 05

src/main/webapp/public/example/example-05.html

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script type="text/javascript">
    $(function() {
        $("body").html("<table border='1' width='300'>" +
            "<tr><th>name</th><th>group</th></tr></table>");

        $.ajax({
            url : "/public/text/example-05.xml",
            success : function(xml) {
                var nations = $(xml).find("nation");
                print(nations);
            }
        });
    });

    function print(nations) {
        nations.each(function(idx, item) {
            var name = $(this).find("name").text();
            var group = $(this).find("group").text();
            var data = "" +
                "<tr align='center'>" +
                "<td>" + name + "</td>" +
                "<td>" + group + "</td>" +
                "</tr>";
        });
    }
}
```

```
        $("table").append(data);
    });
}
</script>
</head>
<body>

</body>
</html>
```

src/main/webapp/public/text/example-05.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<fifa>
  <nation>
    <name>Korea</name>
    <group>B</group>
  </nation>
  <nation>
    <name>Brazil</name>
    <group>A</group>
  </nation>
  <nation>
    <name>Russia</name>
    <group>B</group>
  </nation>
</fifa>
```

테스트

<div><div><div>←</div><div>→</div><div>↺</div><div>🏠</div></div><div>localhost:8080/public/example/example-05.html</div></div>	
name	group
Korea	B
Brazil	A
Russia	B

Example 06

src/main/webapp/public/example/example-06.html

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script type="text/javascript">
    $(function() {
        $.getJSON("/public/text/example-06.json", function(data, textStatus) {
            $.each(data, function(idx, item) {
                $('#target').append(
                    "<tr><td>" + this.day + "</td><td>" + item.title + "</td></tr>");
            });
        });
    });
</script>
</head>
<body>
    <table border="1" id="target"></table>
</body>
</html>
```

src/main/webapp/public/text/example-06.json

```
[
  {
    "day":"2015-01-17",
    "title":"수업시작일"
```

```
    },  
    {  
      "day": "2015-02-08",  
      "title": "오늘"  
    },  
    {  
      "day": "2015-02-19",  
      "title": "설날"  
    }  
  ]  
}
```

테스트

localhost:8080/public/example/example-06.html	
2015-01-17	수업시작일
2015-02-08	오늘
2015-02-19	설날

Example 07

src/main/webapp/public/example/example-07.html

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script type="text/javascript">
    $(function() {
        getSel2('/public/example/example-07.jsp');
    });

    function getSel2(url) {
        var keyword = $('select[name=sel1]').val();

        $.ajax({
            dataType : 'json',
            url : url,
            type : "post",
            data : {
                "keyword" : keyword
            },
            beforeSend : function(xhr) {
                if (xhr && xhr.overrideMimeType) {
                    xhr.overrideMimeType('application/json;charset=utf-8');
                }
            },
            success : displayResult
        });
    }
}
```

```

function displayResult(data) {
    $('select[name=sel2] option').remove();

    $.each(data.result, function(idx, item) {
        $('select[name=sel2]').append(
            '<option value='+item+'>' + item + '</option>');
    });
}
</script>
</head>
<body>
    <div>
        <h2>콤보박스 비동기 처리</h2>
        <form name="search">
            <select name="sel1" onchange="getSel2('/public/example/example-07.jsp')">
                <option value="육류">육류</option>
                <option value="음료">음료</option>
                <option value="패스트푸드">패스트푸드</option>
            </select>
            <select name="sel2" onchange="javascript:alert($('select[name=sel2']).val())">
            </select>
        </form>
    </div>
</body>
</html>

```

src/main/webapp/public/example/example-07.jsp

```
<%@ page language="java" contentType="text/plain; charset=utf-8"%>
<%
    String keyword = request.getParameter("keyword");

    String[] result = null;
    String[] subs1 = { "닭고기", "쇠고기", "돼지고기" };
    String[] subs2 = { "바나나주스", "콜라", "녹차" };
    String[] subs3 = { "라면", "햄버거", "감자튀김" };

    if (keyword.equals("육류")) {
        result = subs1;
    } else if (keyword.equals("음료")) {
        result = subs2;
    } else if (keyword.equals("패스트푸드")) {
        result = subs3;
    } else {
        result = subs1;
    }

    int end = result.length;
    out.print("{W"resultW":[");
    for (int i = 0; i < end; i++) {
        out.print("W"+result[i]+"W");
        if (i < end - 1) {
            out.print(",");
        }
    }
    out.print("]");
%>
```

테스트

← → ↺ ⬆

localhost:8080/public/example/example-07.html

콤보박스 비동기 처리

육류 ▼

닭고기 ▼

← → ↺ ⬆

localhost:8080/public/example/example-07.html

콤보박스 비동기 처리

음료 ▼

바나나주스 ▼

Util 01

폼에 같은 네임을 공유하는 배열이 존재하는 경우 제이쿼리의 `serialize()`, `serializeArray()` 함수로 이를 처리하면 JSON 포맷과 맞지 않으므로 이를 개발자가 수정해야 하는 부담을 안게 된다. 개인 개발자가 오픈소스로 제공하는 `serializeObject()` 함수로 멋지게 처리해 보자.

src/main/webapp/public/util/util-01.html

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script type="text/javascript" src="/public/js/jquery.serializeObject.min.js"></script>
<script type="text/javascript">
    $(function() {
        $("#minutes").submit(function() {
            var minutes = $('form#minutes').serializeObject();
            $("#target").text(JSON.stringify(minutes));
            return false;
        });
    });
</script>
</head>
<body>

    <form id="minutes">
        <div>
            subject <input type="text" name="subject">
        </div>
        <div>
            minute-taker <input type="text" name="minute-taker">
        </div>
```

```

    <div>
      <input type="checkbox" name="attendees" value="David" checked="checked"> David
      <input type="checkbox" name="attendees" value="Daniel" checked="checked"> Daniel
      <input type="checkbox" name="attendees" value="Darwin" checked="checked"> Darwin
    </div>

    <button type="submit">submit</button>
  </form>

  <div id="target"></div>

</body>
</html>

```

src/main/webapp/public/js/jquery.serializeObject.min.js

다운로드 <https://plugins.jquery.com/serializeObject/>

테스트

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/public/util/util-01.html'. The page content includes a form with the following elements:

- A text input field labeled 'subject' containing the value '11'.
- A text input field labeled 'minute-taker' containing the value '22'.
- Three checked checkboxes labeled 'David', 'Daniel', and 'Darwin'.
- A 'submit' button.

Below the form, the serialized JSON object is displayed:

```
{ "subject": "11", "minute-taker": "22", "attendees": ["David", "Darwin"] }
```


Mash-up 01

src/main/webapp/public/mashup/mashup-01.html

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script type="text/javascript">
$(function(){
    $("#btn1").click(function() {
        var data = "";
        var url = "http://apis.daum.net/search/book?"
            + "apikey=28ea9874c487f5e0032ad9b9aed9e58d86fd1527&"
            + "output=json&q=java";
        $.post(
            url,
            function(data, status, xhr) {
                $('#div#view').text(data);
                console.log("Data: " + data + "\nStatus: " + status);
                console.log(xhr.status + " : " + xhr.statusText);
            }
        ).fail(function(xhr, status){
            $('#div#error').text(xhr.statusText);
        });
    });

    $("#btn2").click(function() {
        var page = parseInt($('#page').val(), 10);
        getData(page);
    });
});
```

```

function getData(page){
    console.log("page: " + page);

    // 메소드옵션
    // async : 동기또는비동기를지정(true or false)
    // complete(jqXHR, textStatus) : ajax 완료이벤트핸들러(함수)지정
    // data : 요청매개변수지정(Object, String), 서버쪽으로넘길데이터
    // error(jqXHR, textStatus, errorThrown) : Ajax 실패이벤트핸들러(함수)지정
    // jsonp(JSON with Padding) : JSON 매개변수이름지정
    // jsonCallback : JSONP 콜백함수이름을지정(String, Function)
    // success(data, textStatus, jqXHR) : Ajax 성공이벤트핸들러(함수)지정
    // timeout : 만료시간지정(Number)
    // type : get or post
    // url : 대상 URL
    $.ajax({
        url : "http://apis.daum.net/search/book",
        dataType : "jsonp",
        type : "post",
        jsonp : "callback",
        data : {
            apikey : "28ea9874c487f5e0032ad9b9aed9e58d86fd1527",
            q : "java", // 검색어
            result : "10", // 한페이지에 출력될 결과수
            pageno : page, // 페이지번호
            output : "json" // 결과형식
        },
        success : function(result,status,xhr){
            result = result.channel;
            $("p").text("검색 결과: " + result.totalCount + "개");

            $('table tr').remove();
            var title = "";

```

```

        $.each(result.item, function(idx, data){
            title = data.title;
            title = title.replace("<b>", "[");
            title = title.replace('</b>', ']');
            var str = "<tr><td>" + title + "</td></tr>";

            $('table').append(str);
        });

        //$('#div#view').text(JSON.stringify(result));
    },
    error : function(xhr,status,error){
        console.log(xhr.status + " : " + xhr.statusText);
        console.log(status);
        console.log(error);
    }
});
}

$('#prev').click(function(){
    var page = parseInt($('#page').val(), 10);

    if (page > 1) {
        page = page - 1;
    }

    $('#page').val(page);
    getData(page);
});

$('#next').click(function(){
    var page = parseInt($('#page').val(), 10);

```

```

        page = page + 1;

        $('#page').val(page);
        getData(page);
    });
});
</script>
</head>
<body>

    <label><button id="btn1">get then fail</button></label>
    <label><button id="btn2">get then success</button></label>
    <div id="error"></div>
    <hr>

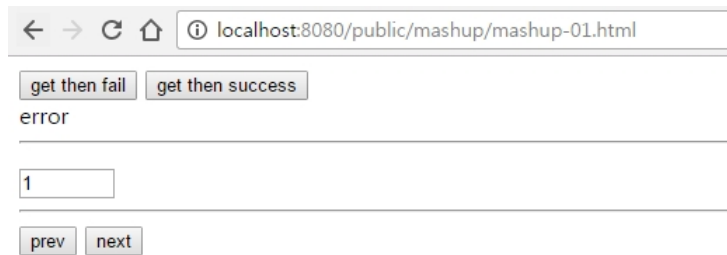
    <div id="view">
        <p></p>
        <table></table>
        <input type="text" id="page" name="page" value="1"
            size="5" readonly="readonly">
        <hr>

        <button id="prev">prev</button>
        <button id="next">next</button>
    </div>

</body>
</html>

```

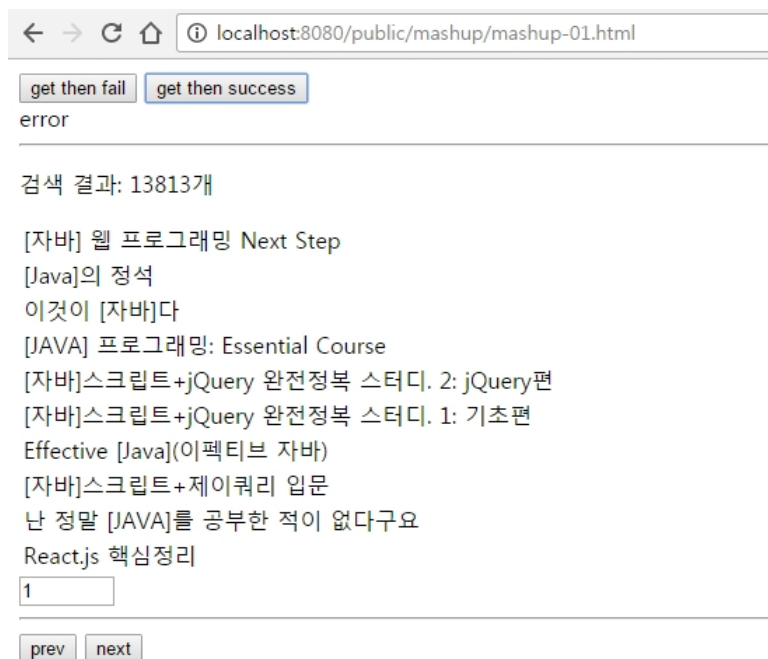
테스트



버튼 "get then fail"을 눌렀을 때 발생하는 에러 메시지

XMLHttpRequest cannot load http://apis.daum.net/search/book
?apikey=28ea9874c487f5e0032ad9b9aed9e58d86fd1527&output=json&q=java.
No 'Access-Control-Allow-Origin' header is present on the requested resource.
Origin 'http://localhost:8080' is therefore not allowed access.

"get then success" 버튼을 누르면 성공적으로 데이터를 가져온다.



Mash-up 02

src/main/webapp/public/mashup/mashup-02.html

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script type="text/javascript">
    $(function() {
        $.getJSON(
            "http://ax.itunes.apple.com/" +
            "WebObjects/MZStoreServices.woa/ws/RSS/topsongs/limit=10/json",
            function(data) {
                var feed = data.feed;
                var entry = feed.entry;

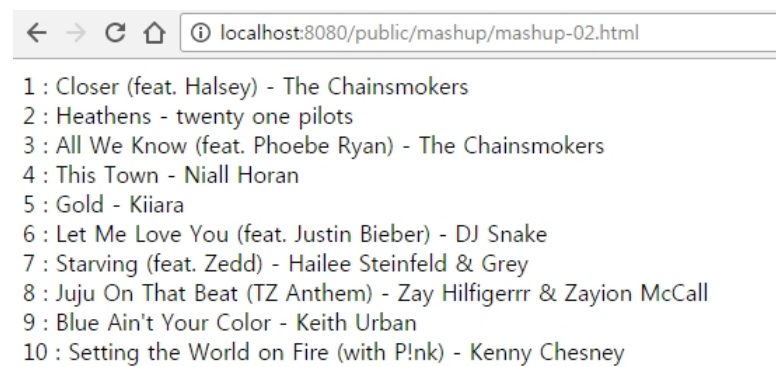
                console.log(feed);
                console.log(entry);

                $.each(entry, function(idx, item) {
                    var title = item.title;
                    $('#target').append(
                        (idx + 1) + ' : ' + title.label + '<br>');
                });
            });
    });
</script>
</head>
<body>
```

```
<div id="target"></div>

</body>
</html>
```

테스트



CORS 설정

참고: <https://spring.io/guides/gs/rest-service-cors/>

새 프로젝트 만들기

STS > Spring Boot Project > 디펜던시 web 선택

프로젝트명: javascript-ajax-cors

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>rest-service-cors</artifactId>
  <version>0.1.0</version>
  <packaging>jar</packaging>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.4.1.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
```



```
<java.version>1.8</java.version>
</properties>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>

</project>
```

application.properties

```
server.port=8090
```

Greeting.java

```
package com.example.common.model;

public class Greeting {
    private final long id;
    private final String content;

    public Greeting() {
        this.id = -1;
        this.content = "";
    }

    public Greeting(long id, String content) {
        this.id = id;
        this.content = content;
    }

    public long getId() {
        return id;
    }

    public String getContent() {
        return content;
    }
}
```

GreetingController.java

```
package com.example.common.controller;

import java.util.concurrent.atomic.AtomicLong;
```

```

import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.example.common.model.Greeting;

@RestController
//enables cross-origin requests for methods in this class
@CrossOrigin(origins = "http://localhost:8080")
public class GreetingController {

    private static final String template = "Hello, %s!";
    private final AtomicLong counter = new AtomicLong();

    // shortcut for @RequestMapping(method = RequestMethod.GET)
    @GetMapping("/greeting")
    public Greeting greeting(
        @RequestParam(required=false, defaultValue="World") String name) {
        System.out.println("==== in greeting =====");
        return new Greeting(counter.incrementAndGet(), String.format(template, name));
    }
}

```

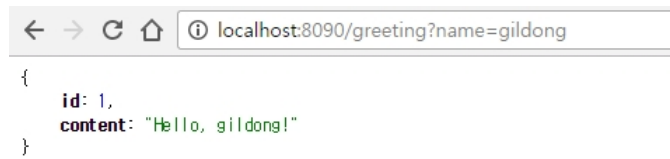
배포파일 생성

프로젝트 > Run As > Maven install

배포파일로 서비스 시작

콘솔 > 프로젝트 루트폴더로 이동 > java -jar target/rest-service-cors-0.1.0.jar

테스트



기존 프로젝트를 통해 CORS 검증

프로젝트 javascript-ajax-2 를 가지고 검증을 위한 작업을 진행한다.

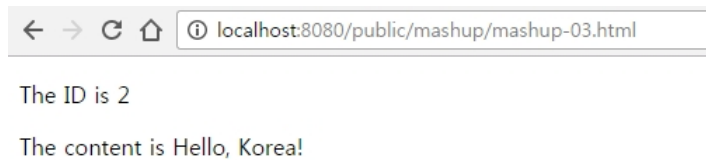
src/main/webapp/public/mashup/mashup-03.html

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"> </script>
<script type="text/javascript">
    $(function() {
        $.ajax({
            url : "http://localhost:8090/greeting?name=Korea"
        }).then(function(data, status, jqxhr) {
            $('#greeting-id').append(data.id);
            $('#greeting-content').append(data.content);
            console.log(jqxhr);
        });
    });
</script>
</head>
<body>
```

```
<div>
  <p class="greeting-id">The ID is </p>
  <p class="greeting-content">The content is </p>
</div>

</body>
</html>
```

테스트



← → ↺ ⬆ ⓘ localhost:8080/public/mashup/mashup-03.html

The ID is 2

The content is Hello, Korea!

서버 포트를 변경하고 다시 기동하여 앞서서 사용한 URL 로 재 접속하여 검증해 보자.

application.properties

```
server.port=9000
```



← → ↺ ⬆ ⓘ localhost:9000/public/mashup/mashup-03.html

The ID is

The content is

에러 메시지 확인

```
MLHttpRequest cannot load http://localhost:8090/greeting?name=Korea. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://localhost:9000' is therefore not allowed access. The response had HTTP status code 403.
```

포트를 다시 8080 로 바꾼 후 테스트 하면 처리가 된다.

전역 CORS 설정

프로젝트: javascript-ajax-cors

JavascriptAjaxCorsApplication.java

```
package com.example;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@SpringBootApplication
public class JavascriptAjaxCorsApplication {

    public static void main(String[] args) {
        SpringApplication.run(JavascriptAjaxCorsApplication.class, args);
    }

    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurerAdapter() {
            @Override
            public void addCorsMappings(CorsRegistry registry) {
                registry.addMapping("/say/**")
                    .allowedOrigins("http://localhost:8080");
            }
        };
    }
}
```

CorsRegistry 설정옵션 설명

```
registry.addMapping("/")
/* * 는 모든 도메인에 대해 허용하겠다는 의미이다.
.allowedOrigins("*")
// 특정 도메인에만 허용할 수 있다.
.allowedOrigins("http://www.company.co.kr")
.allowedOrigins("http://test.company.co.kr")
// POST, GET, PUT, DELETE, OPTIONS 요청에 대해 허용한다.
.allowedMethods("POST", "GET", "PUT", "DELETE", "OPTIONS")
/*
* JQuery AJAX 요청 시 헤더(x-requested-with)를 포함한다.
* 요청이 Ajax 요청임을 알려주기 위해 Header 에 x-request-width 를 설정한다.
* 폼을 통한 요청과 Ajax 요청을 구분하기 위해 사용되는 비표준 규약이다.
* 많은 라이브러리에서 이를 채택하여 사용하고 있다.
* HTML5 부터는 폼과 Ajax 요청을 구분할 수 있는 Header 가 추가되었다.
*/
.allowedHeaders("x-requested-with")
// 사용자의 계정을 사용하여 접근하는 것을 금지한다.
.allowCredentials(false)
/*
* HTTP Request 요청에 앞서 Preflight Request 를 사용하여
* 해당 서버에 요청하는 메서드가 실행 가능한지(권한이 있는지) 확인한다.
* Preflight Request 는 OPTIONS 메서드를 통해 서버에 전달된다.
* Access-Control-Max-Age 는 Preflight request 를 캐시할 시간이다.
* 단위는 초단위이며, 3600 초는 1 시간이다.
* 최소 1 시간동안 서버에 재 요청하지 않을 것이다.
*/
.maxAge(3600);
```

GreetingController2.java

```
package com.example.common.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

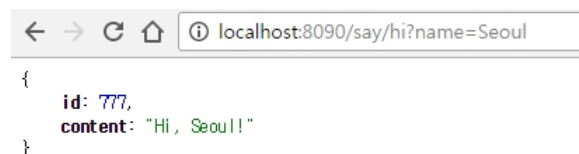
import com.example.common.model.Greeting;

@RestController
@RequestMapping("/say")
public class GreetingController2 {

    private static final String template = "Hi, %s!";

    @GetMapping("/hi")
    public Greeting greeting(
        @RequestParam(required = false, defaultValue = "World") String name) {
        return new Greeting(777, String.format(template, name));
    }
}
```

테스트



The screenshot shows a web browser window with the address bar displaying `localhost:8090/say/hi?name=Seoul`. Below the address bar, a JSON response is shown in a light blue box:

```
{
  id: 777,
  content: "Hi, Seoul!"
}
```


프로젝트: javascript-ajax-2 에 검증파일 추가

src/main/webapp/public/mashup/mashup-04.html

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
<script type="text/javascript">
    $(function() {
        $.ajax({
            url : "http://localhost:8090/say/hi?name=Seoul"
        }).then(function(data, status, jqxhr) {
            $('#greeting-id').append(data.id);
            $('#greeting-content').append(data.content);
            console.log(jqxhr);
        });
    });
</script>
</head>
<body>

    <div>
        <p class="greeting-id">The ID = </p>
        <p class="greeting-content">The content = </p>
    </div>

</body>
</html>
```

테스트

← → ↻ ⬆ ⓘ localhost:8080/public/mashup/mashup-04.html

The ID = 777

The content = Hi, Seoul!

Filter 를 사용한 CORS 설정

프로젝트: javascript-ajax-cors

CORSFilter.java

```
package com.example.common.filter;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;

@Component
@Order(Ordered.HIGHEST_PRECEDENCE)
public class CORSFilter implements Filter {

    private final Logger log = LoggerFactory.getLogger(CORSFilter.class);

    @Override
    public void destroy() {
```

```

    }

    @Override
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
        throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) req;
        HttpServletResponse response = (HttpServletResponse) res;

        response.setHeader("Access-Control-Allow-Origin", "http://localhost:8080");
        response.setHeader("Access-Control-Allow-Methods",
            "POST, PUT, GET, OPTIONS, DELETE");
        response.setHeader("Access-Control-Max-Age", "3600");
        response.setHeader("Access-Control-Allow-Headers",
            "Content-Type, Accept, X-Requested-With, X-Auth-Token, remember-me");
        response.setHeader("Access-Control-Allow-Credentials", "false");

        if (!"OPTIONS".equalsIgnoreCase(request.getMethod())) {
            chain.doFilter(req, res);
        }
    }

    @Override
    public void init(FilterConfig config) throws ServletException {
        log.info("CORSFilter init() called");
    }
}

```

스프링이 제공하는 CORS 설정을 주석처리한다.

1. GreetingController > @CrossOrigin
2. JavascriptAjaxCorsApplication > corsConfigurer()

두 개의 프로젝트를 기동한다.

1. javascript-ajax-2
2. javascript-ajax-cors

테스트

```
← → ↺ ⬆ localhost:8090/greeting
{
  id: 1,
  content: "Hello, World!"
}
```

```
← → ↺ ⬆ localhost:8080/public/mashup/mashup-03.html

The ID is 2

The content is Hello, Korea!
```

```
← → ↺ ⬆ localhost:8080/public/mashup/mashup-04.html

The ID = 777

The content = Hi, Seoul!
```