

jQuery

Cos'è jQuery

Perché jQuery

"Write less. Do more"

Caratteristiche

Uso di jQuery

Selettori CSS

Esempi:

Selettori generici

Metodi Core

Manipolazione del Wrapped Set

get()

toArray()

not()

each()

Manipolazione degli attributi

attr()

removeAttr()

Manipolazione delle classi

addClass()

removeClass()

toggleClass()

hasClass()

Manipolazione degli stili

css()

Manipolazione dei contenuti

html()

text()

Animazioni

show()

hide()

toggle()

Manipolazione dei contenuti

append()

Gestione degli Eventi

Eventi

JavaScript non-Intrusivo

Definizione di Event Handler

Esempio jQuery#10

jQuery e AJAX

Interazione Client-Server

Interazione Asincrona : AJAX

XML

Interazione AJAX

Interazione AJAX con jQuery

Esempio jQuery#12

Codifica JSON

AJAX JSON

Esempio jQuery#13

Cos'è jQuery

- **jQuery core:** libreria JavaScript nella quale sono definiti operatori per:
 - La selezione di parti di un documento;
 - La manipolazione degli elementi del DOM;
 - La gestione degli eventi;
 - La definizione di eventi visuali sugli elementi del documento;
 - La gestione dell'interazione con il server tramite AJAX

Perché jQuery

"Write less. Do more"

- **Senza jQuery:**

```
VAR TABLE = DOCUMENT.GETELEMENTSBYTAGNAME('TABLE');
for(VAR t = 0; t < TABLE.LENGTH; t++){
    VAR rows = TABLE[t].GETELEMENTSBYTAGNAME('tr');
    for (VAR i = 1; i < rows.length; i +=2) {
        rows[i].CLASSNAME += 'striped';
    }
}
```

Con la variabile *TABLE* associo il risultato dell'esecuzione del metodo `getElementsByName` al quale passiamo il parametro *table*. Questo metodo estrae nel DOM tutti gli elementi con il tag coincidente con il parametro passato. Alla variabile *table* associamo tutti gli oggetti di tipo tabella che la nostra pagina ha

e nel for usiamo `t < TABLE.LENGTH` perché indica il numero degli elementi contenuti in table. Nel corpo della for associamo a rows il risultato della get.. applicato al t-esimo dell'array table ovvero la t-esima tabella della nostra pagina. Quindi per ogni tabella estratta estraiamo il numero di righe che la tabella contiene.

Adesso con il secondo for (che stavolta incrementa di 2 ogni volta) prendiamo l'i-esima riga della t-esima tabella e per ogni riga dispari (dato che incrementa di 2 ogni volta) associamo al suo attributo `className` aggiungiamo la stringa `'striped'`.

- **Con jQuery:**

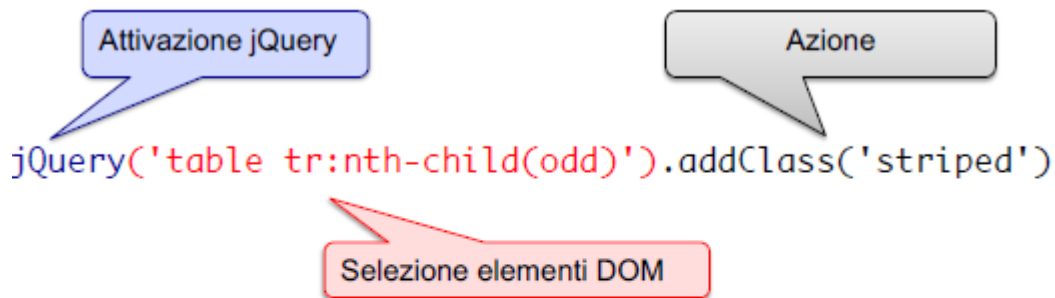
```
$('#TABLE tr:nth-child(odd)').ADDCLASS('striped');
```

- **jQuery UI:** libreria di componenti scritta in jQuery core articolata in categorie (che non vedremo) tra le quali:
 - Effetti (in aggiunta a quelli del core)
 - Interazioni (drag & drop, sorting, ...)
 - Widgets (progress bar, sldiers, dialog box, ...)

Caratteristiche

- Primo rilascio: 2006
- Piccole dimensioni. ~30Kb nella versione minima compressa
- Ricca di selettori HTML DOM veloci e di facile uso
- Compatibile con tutti i browser principali. Elimina problemi di codifica cross-browser
- È open-source
- È ben documentata
- È estensibile. Continuamente arricchita di componenti (plugin)
- È integrabile con altre librerie analoghe (Prototype, MooTools)
- È una delle librerie più utilizzate JavaScript

Uso di jQuery



- La chiamata alla funzione jQuery **ritorna un oggetto JavaScript (wrapped set) che contiene:**
 - un array con gli elementi del DOM selezionati, nell'ordine in cui compaiono nel documento;
 - una serie di metodi predefiniti che operano sull'oggetto

- Notazione alternativa per la chiamata: jQuery → \$

```
$('#table tr:nth-child(even)').addClass('striped')
```

- **"Method chaining"**: la maggior parte dei metodi jQuery (wrapped methods) ritorna come risultato gli stessi elementi ricevuti in input. Questo permette l'attivazione in cascata di azioni sugli elementi selezionati

```
$('#table tr:nth-child(even)').hide().addClass('removed')
```

- Attivazione di jQuery: al termine del caricamento del DOM

```
$(document).ready(function(){<Codice jQuery>})  
//oppure  
$(function(){<Codice jquery>})
```

Selettori CSS

| Selettore | Descrizione |
|-----------|---|
| * | Tutti gli elementi |
| E | Elementi con tag E |
| E F | Elementi con tag F discendenti di elemento con tag E |
| E>F | Elementi con tag F figli diretti di elemento con tag E |
| E.F | Elementi con tag E e classe F |
| E#I | Elementi con tag E e id I |
| E[A] | Elementi con tag E e attributo A (senza condizioni sul valore) |
| E[A=V] | Elementi con tag E e attributo A di valore V |
| E[A^=V] | Elementi con tag E e attributo A con valore che inizia con V |
| E[A\$=V] | Elementi con tag E e attributo A con valore che termina con V |
| E[A!=V] | Elementi con tag E e attributo A di valore diverso da V o senza A |
| E[A*=V] | Elementi con tag E e attributo A con valore che contiene V |

Esempi:

| Esempio | Selezione |
|--|--|
| <code>\$('div, span')</code> | Gli elementi 'div' o 'span' del documento |
| <code>\$('ul li img')</code> | Le immagini negli item delle liste non ordinate |
| <code>\$('ul>li img')</code> | Le immagini negli item di primo livello delle liste non ordinate |
| <code>\$('ul.miaClasse')</code> | Le liste non ordinate con classe miaClasse |
| <code>\$('#intestazione')</code> | Gli elementi con id='intestazione' |
| <code>\$('form[method]')</code> | Le form con l'attributo method definito |
| <code>\$('input[type=checkbox][checked]')</code> | Gli elementi input di tipo checkbox selezionati |
| <code>\$('img[src^='dog']')</code> | Le immagini con valore di src che inizia per 'dog' |
| <code>\$('img[src\$='dog']')</code> | Le immagini con valore di src che finisce per 'dog' |
| <code>\$('img[src!='dog']')</code> | Le immagini con valore di src diverso da 'dog' |
| <code>\$('img[src*='dog']')</code> | Le immagini con valore di src che contiene 'dog' |

Selettori generici

| Selettore | Descrizione |
|----------------|---|
| :button | Elementi di tipo button, submit, reset |
| :input | Elementi di una form (input, select, textarea, button) |
| :hidden | Elementi hidden |
| :header | Elementi h1...h6 |
| :not(selector) | Elementi non corrispondenti al selector |
| :selected | Elementi option selezionati |
| :submit | Elementi <input type="submit"> o <button type="submit"> |
| :visible | Elementi visibili |
| ... | ... |

Metodi Core

Manipolazione del Wrapped Set

get()

`get(index)` : ritorna l'elemento di posizione index nel set a cui è applicato. Se index è omissso, il risultato è l'intero set

```
var imgElement = $('img[alt]').get(0) //estraiamo la prima im
```

toArray()

`TOARRAY()` : crea un array con tutti gli elementi del set a cui è applicato

```
$('#div').toArray()
```

not()

`not(expression)` : crea una copia del wrapped set a cui è applicato ed elimina da essa tutti gli elementi che corrispondono ai criteri definiti da expression

- `$('#img').not("[alt='ELIMINAMI']")`
- `$('#img').not(function(){return !$(this).hasClass('tienimi`

each()

EACH(ITERATOR) : scandisce tutti gli elementi del wrapped set a cui è applicato ed attiva su ognuno la funzione ITERATOR

```
$('#img').each(function(n){ //function n sarebbe l'iesimo ele  
this.alt="QUESTA è L'IMMAGINE n. " + n + " con id=" + this.id
```

Manipolazione degli attributi

attr()

ATTR(NAME) : ritorna il valore dell'attributo NAME del primo elemento del set a cui è applicato o undefined se l'attributo è assente

ATTR(NAME, VALUE) : assegna il valore value all'attributo name di tutti gli elementi set a cui è applicato

```
- $('#myImage').attr('src')  
- $(':input').attr('title', 'Inserisci un VALORE')
```

removeAttr()

removeAttr(NAME) : rimuove l'attributo name da tutti gli elementi del set a cui è applicato

```
 $(':input').removeAttr('TARGET')
```

Manipolazione delle classi

addClass()

ADDCLASS(NAMES) : aggiunge la classe (o le classi) NAMES agli elementi del set a cui è applicato. NAMES può essere una funzione

```
 $('#:img').addClass('mediumSize','highlight')
```

removeClass()

removeClass(names) : rimuove la classe (o le classi) names agli elementi del set a cui è applicato. names può essere una funzione

```
$(':img').removeClass('mediumSize')
```

toggleClass()

`toggleClass(names)` : aggiunge la classe (o le classi) names agli elementi del set a cui è applicato che non la posseggono e la rimuove da quelli che la posseggono. names può essere una funzione

```
$(':img').toggleClass('mediumSize')
```

hasClass()

`hasClass(name)` : ritorna true se tutti gli elementi del set a cui è applicato posseggono la classe name, false altrimenti

```
$(':img').hasClass('mediumSize')
```

Manipolazione degli stili

css()

`css(NAME, VALUE)` : assegna alla proprietà di stile NAME il valore VALUE per tutti gli elementi del set a cui è applicato

`css(NAME)` : ritorna il valore VALUE della proprietà di stile NAME del primo elemento del set a cui è applicato

`css(properties)` : assegna le proprietà properties a tutti gli elementi del set a cui è applicato

```
- $('div.epandable').css('width', '500')  
- $('div.epandable').css('width')  
- $('div.epandable').css({  
  width: '500',  
  height: '300', backgroundColor: 'white'  
})
```

Manipolazione dei contenuti

html()

`html()` : estrae il codice HTML contenuto nel primo elemento del set a cui è applicato

`html(content)` : assegna il codice HTML in content a tutti gli elementi del set a cui è applicato

```
<ul id="lista">
<li>Uno</li>
<li>Due</li>
<li>Tre</li>
</ul>
-$('#lista').html() --> '<li>Uno</li><li>Due</li><li>Tre</li>'
<ul id="lista">
-$('#LISTA').HTML('<LI>quattro</LI>') -->    <LI>QUATTRO</LI>
</ul>
```

text()

`text()` : estrae il testo contenuto negli elementi del set a cui è applicato

`text(content)` : assegna il testo contenuto in content a tutti gli elementi del set a cui è applicato. I caratteri < > & vengono convertiti in codici UNICODE

```
<ul id="lista">
<li>Uno</li>
<li>Due</li>
<li>Tre</li>
</ul>
-$('#lista').text() --> 'UnoDueTre'
<ul id="LISTA">
-$('#lista').text('<li>quattro</li>') -->    &lt;li&gt;quattro
</ul>
```

Animazioni

show()

`show(speed, callback)` : rende visibile ogni elemento hidden del set a cui il metodo è applicato. `speed` definisce la velocità dell'effetto (in ms. o `slow`, `normal`, `fast`) e `callback` la funzione da attivare al termine dell'animazione

hide()

`hide(speed, callback)` : opera in maniera analoga a `show()`, rendendo invisibili gli elementi a cui è applicato

toggle()

`toggle(speed, callback)` : attiva il metodo `show()` sugli elementi hidden e quello `hide()` sugli elementi non-hidden del wrapped set

Manipolazione dei contenuti

append()

`append(content)` : opera come `html()`, con la differenza che il codice html in `content` viene aggiunto a quello degli elementi del set a cui il metodo è applicato. Il parametro può essere anche una funzione, alla quale vengono passati l'indice e il contenuto precedente di ogni elemento del set

- `$('#p').append('un qualsiasi testo')`
- `$('#p.appendToMe').append($('#A.APPENDME'))` - equivale ad un.
- `$('#p.appendToMe').append(someElement)`

Gestione degli Eventi

Eventi

Gli *eventi* sono azioni generate dall'utente o dal browser durante la visualizzazione di un documento.

Modello operativo del Browser :



La reazione agli eventi viene gestita dagli **event handlers** :

- Definiti nel Browser (attivazione di ancore, submit di form)
- Definiti da programmi (JavaScript) e codificati nei documenti

Al verificarsi di un evento, un oggetto con le informazioni ad esso relative (elemento del documento che l'ha generato, tasto premuto ...) viene passato a tutti gli handler definiti per l'evento, attivati in una sequenza che rispecchia la gerarchia di definizione (*event bubbling*). Vengono attivati tutti gli handler dell'evento definiti negli elementi del DOM che contengono quello in cui l'evento si è verificato.

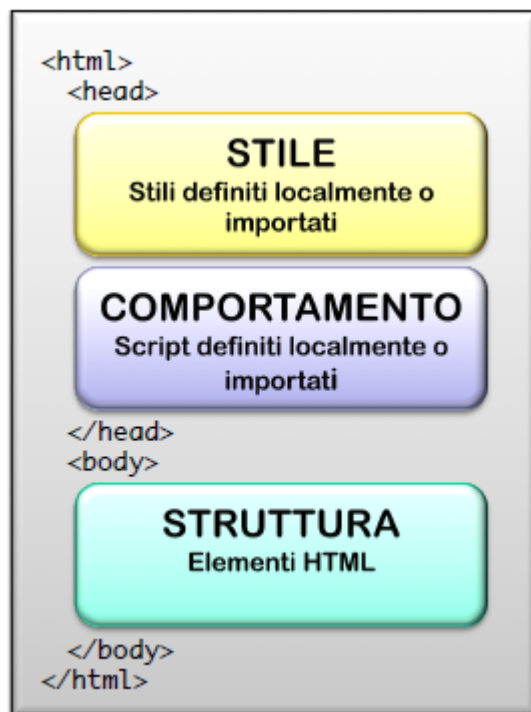
Problema : ogni browser ha un suo modello degli eventi, che determina modalità diverse di gestione da parte degli event handlers (diversa implementazione delle specifiche W3C del DOM Livello 2 tra Internet Explorer e tutti gli altri browser).

Soluzione : usare una libreria che fornisce funzionalità cross-browser per la definizione degli event handlers.

JavaScript non-Intrusivo

Problema : separare la definizione di un elemento della pagina HTML dal codice JS che ne descrive il comportamento (analogia con HTML - CSS).

Soluzione : localizzare tutto il codice JS nella sezione HEAD del documento, selezionando dinamicamente gli elementi HTML ai quali applicarlo.



Definizione di Event Handler

| on() | |
|--|---|
| on(event,selector,data,handler) | |
| Definisce la funzione di data handling dell'evento indicato per tutti gli elementi del set selezionato | |
| event | Nome dell'evento a cui associare l'handler (blur, change, click, focus, ...) |
| selector | Parametro opzionale che seleziona gli elementi figli (nel DOM) di quelli a cui il metodo on() è applicato che attivano l'handler |
| data | Parametro opzionale passato all'event handler attraverso la proprietà data dell'oggetto Event |
| handler | Funzione che implementa l'event handler. L'istanza dell'evento che la attiva viene passato come parametro alla funzione |

| hover() | |
|--|--|
| <code>hover(enterHandler, leaveHandler)</code> | Definisce le azioni da attivare al verificarsi degli eventi <code>mouseenter</code> e <code>mouseleave</code> sugli elementi a cui è applicato |

- **Definizione sintetica di event handlers** (non permette il passaggio del parametro `data`):
 - `blur(handler)`, `change(handler)`, `click(handler)`, `focus(handler)` ...
- **Proprietà dell'oggetto event**:
 - `altKey`, `ctrlKey`, `currentTarget`, `data`, `metaKey`, `pageX`, `pageY`, `relatedTarget`, `screenX`, `screenY`, `shiftKey` ...

Esempio jQuery#10

Il risultato che vogliamo ottenere è un processo che, all'interno di una pagina che propone all'utente una form di inserimento dati, l'utente è obbligato ad inserire i dati secondo dei vincoli.

Se questi vincoli non sono soddisfatti, il client deve NON inviare i dati al server e deve segnalare gli errori.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Validazione Form con jQuery</title>
    <script type="text/javascript" src="src/jquery.js">
    </script>
    <style>
      ._error {
        background-color: #f3e4e4;
        border: solid 2px #ff0000;
      }
    </style>
    <script type="text/javascript">
      $(function () {
        /* Seleziona tutti gli elementi di input contenuti nella pagi
        e setta un handler per l'evento change di uno qualunque degli
```

```

di input della nostra pagina, il quale attiverà una funzione
        $(':input').on('change', function (event) {
// Definisce nella variabile element l'input su cui si è verificato
        var element = $(this);
// Rimuove sull'elemento su cui si è effettuato il click la classe _error
        element.removeClass('_error');
        switch (element.attr('class')) {
            case '_user':
/* Una variabile di tipo pattern è contenuto all'interno di /
L'inizio della stringa viene definito tramite ^ . */
                var pattern = /^[A-Za-z0-9_\-\.\%]{6,}$/;
/* Il metodo test() prende come parametro una stringa e ritorna true
se la stringa passata gli come parametro rispetta le regole definite
nel pattern */
                if (!pattern.test(element.val()))
                    element.addClass('_error');
            }
            break;
            case '_passwd':
                if (element.val().length < 6) {
                    element.addClass('_error');
                }
                break;
            case '_email':
                var pattern = /^[A-Za-z0-9_\-\.\%]{6,}$/;
                if (!pattern.test(element.val()))
                    element.addClass('_error');
            }
            break;
        }
    });

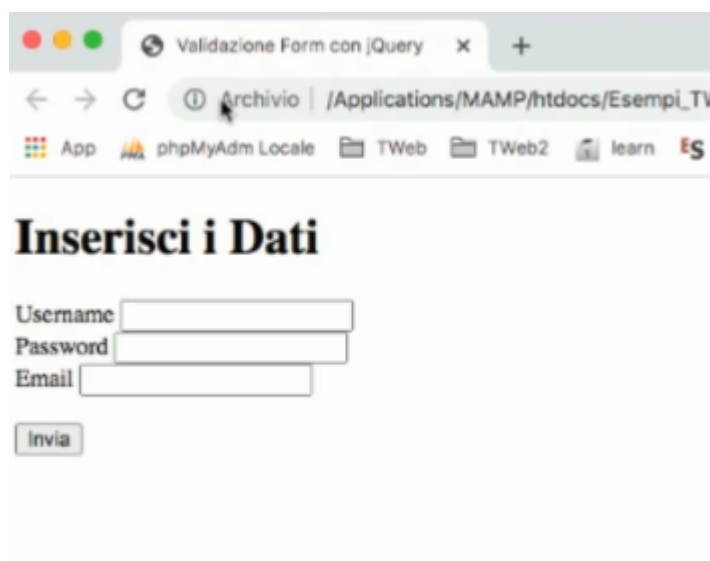
// Handler per la submit
    $('form').on('submit', function (event) {
/* Il metodo trigger attiva gli eventi che gli passo come parametro
(in questo caso change) su tutti gli elementi selezionati */
        $(':input').trigger('change');
    });

```

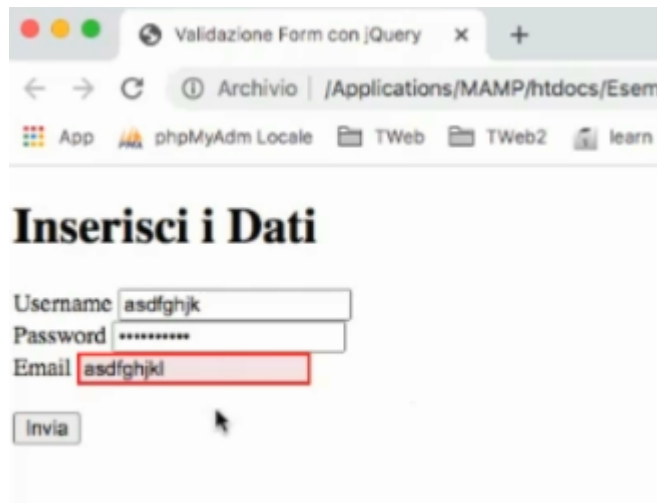
```

/* Estraggo (filtro) solo gli elementi di input che hanno un
class che contiene la classe _error. Se il numero di questi e
è diverso da zero, significa che almeno un elemento non ha pa
la validazione (almeno un elemento ha generato la condizione
        if ($(':input').filter('[class*=_error]')
            return false;
        }
    };
});
});
</script>
</head>
<body>
    <h1>Inserisci i Dati</h1>
    <form action="" method="post">
        <label>Username <input type="text" name="user" id="
        <label>Password <input type="password" name="pass
        <label>Email <input type="text" name="email" id="
        <input type="submit" value="Invia">
    </form>
</body>
</html>

```



The screenshot shows a web browser window with the title 'Validazione Form con jQuery'. The address bar shows the path '/Applications/MAMP/htdocs/Esempi_TV'. The browser's bookmark bar includes 'App', 'phpMyAdm Locale', 'TWeb', 'TWeb2', 'learn', and 'ES'. The main content of the page is a form titled 'Inserisci i Dati' in a large, bold, black serif font. Below the title, there are three input fields: 'Username' (a text input), 'Password' (a password input), and 'Email' (a text input). At the bottom of the form is a button labeled 'Invia'.



- Metodi analoghi:
 - `prepend(content)`, `before(content)`, `after(content)`

remove()

`remove(selector)` : rimuove gli elementi del set a cui il metodo è applicato. `selector` definisce un ulteriore selettore di un sottoinsieme del set da eliminare

```
$( 'p' ).remove()  
$( 'p' ).remove( '.miaClass' )
```

replaceWith()

`replaceWith(content)` : sostituisce gli elementi del set a cui il metodo è applicato con `content`.

```
$( 'img[alt]' ).each( function() {  
    $( this ).replaceWith( '<div>' + $( this ).attr( 'alt' ) + '</div>' );  
});
```

Valori delle Form

val()

`val()` : ritorna il valore dell'attributo `VALUE` del primo elemento del set a cui è applicato. Se è un elemento `select`, ritorna il valore dell'elemento

selezionato, o un array di valori per select multipli.

VAL(VALORE) : assegna VALORE all'attributo VALUE di tutti gli elementi del set a cui è applicato. VALUE può essere una funzione

VAL(VALUES) : marca come selezionati tutti gli elementi checkbox, RADIO e option del set a cui è applicato se il loro valore corrisponde ad uno degli valori dell'array VALUES

- `$('[name="radiogroup"]:checked').val()`
- `$('[name="radiogroup"]:checked').val('set')`
- `$('input,select').VAL(['uno','due','tre'])`

Animazioni

| fadeIn() | |
|--------------------------------------|---|
| <code>fadeIn(speed, callback)</code> | Opera in maniera analoga a <code>show()</code> , usando un effetto di dissolvenza |

| fadeOut() | |
|---------------------------------------|---|
| <code>fadeOut(speed, callback)</code> | Opera in maniera analoga a <code>hide()</code> , usando un effetto di dissolvenza |

jQuery e AJAX

Interazione Client-Server

JavaScript consente di modificare il contenuto di un documento visualizzato dal browser tramite manipolazione del DOM.

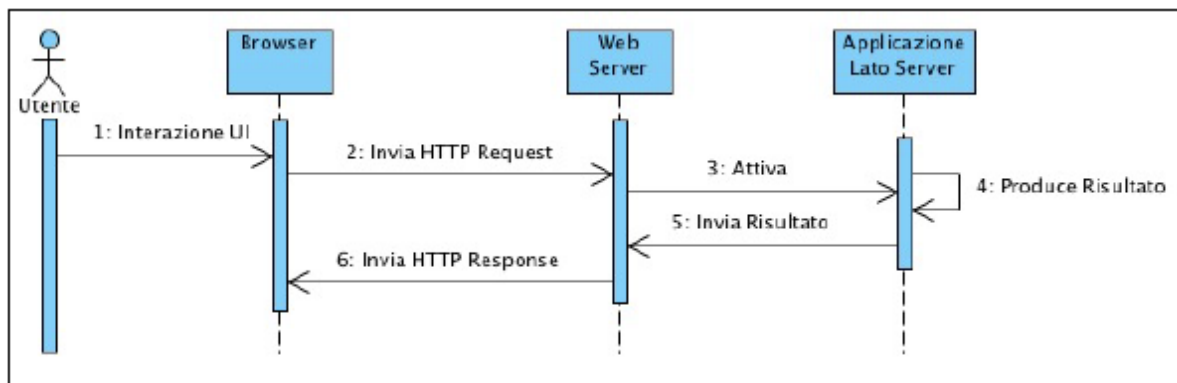
Possiamo iniettare in un documento informazioni provenienti dal server a seguito di una richiesta (GET o POST) fatta dal client ?

NO , usando il protocollo HTTP : ad ogni richiesta del client corrisponde un processo di acquisizione di un intero documento HTML che viene visualizzato al posto del precedente.

Il protocollo HTTP è **sincrono** : ad ogni richiesta il client aspetta che il server risponda per poi visualizzare la risposta; nell'attesa, nessun'altro evento viene

rilevato (e gestito) dal browser.

HTTP Sequence Diagram



Interazione Asincrona : **AJAX**

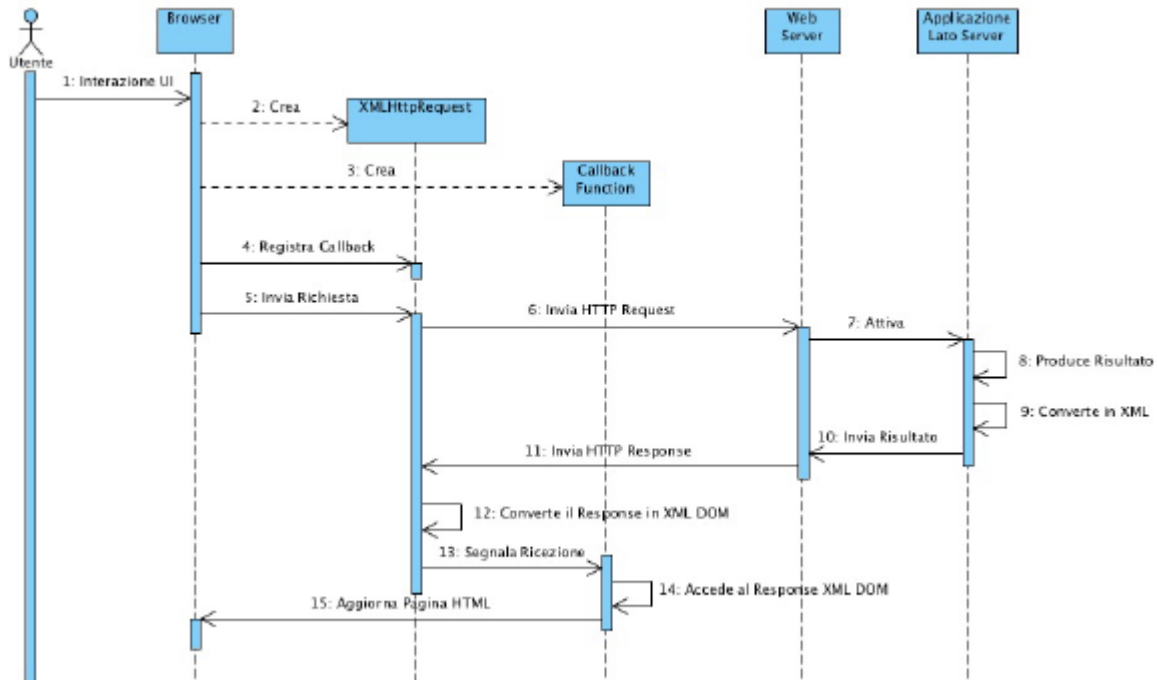
- 1998 : Microsoft introduce un modello di comunicazione asincrona tra client e server basata sulla definizione di un nuovo componente ActiveX per la sua applicazione Outlook Web Access (OWA).
- 2005 : un modello di processo per comunicazione *asincrona* viene standardizzato e reso pubblico : **AJAX** (*Asynchronous JavaScript and XML*).
 - Interazione C/S gestita tramite un oggetto della classe

`XMLHttpRequest`

- Dati scambiati utilizzando, di norma, la codifica XML

Risultato : possibilità di fare una richiesta al server ed acquisire dati che possono essere iniettati nel documento visualizzato senza doverlo ricaricare.

AJAX Sequence Diagram



XML

L'**XML** (*eXtensible Markup Language*) è un meta-linguaggio derivato dall'SGML e sviluppato a partire dal 1996 per strutturare documenti. I Tag XML hanno stessa struttura di quelli definiti nell'HTML (sono parole racchiuse tra `<>` ed ammettono attributi con sintassi `nome="valore"`). A differenza dell'HTML, i Tag XML non sono predefiniti, e quindi la loro interpretazione non è definita all'interno del linguaggio. I Tag non definiscono in alcun modo il formato di visualizzazione dell'oggetto che racchiudono.

Esempio :

```

<?xml version="1.0" encoding="UTF-8"?>
<oggetto>
  <province regione="Marche">
    <provincia>Ascoli Piceno</provincia>
    <provincia>Ancona</provincia>
    <provincia>Fermo</provincia>
    <provincia>Pesaro Urbino</provincia>
    <provincia>Macerata</provincia>
  </province>
  <province regione="Abruzzo">
    <provincia>Chieti</provincia>
    <provincia>L'Aquila</provincia>
    <provincia>Pescara</provincia>
    <provincia>Teramo</provincia>
  </province>
</oggetto>

```

Interazione AJAX

Basata su due moduli software :

1. L'oggetto istanza della classe `XMLHttpRequest` lato client
2. Un programma lato server che acquisisce la richiesta e fornisce i dati nel formato richiesto (XML)

Processo lato client :

- Creazione istanza `XMLHttpRequest`
- Inizializzazione dell'oggetto : metodo di comunicazione (GET o POST), definizione del modulo da attivare sul server, definizione della funzione di callback per l'elaborazione dei dati provenienti dal server
- Invio della richiesta al server

Esempio jQuery#11

Abbiamo una pagina che presenta una form con due select : una che ci consente di selezionare una regione e l'altra le relative province.

```

<!DOCTYPE html>

<html>
  <head>

```

```

<meta charset="UTF-8">
<title>Esempio AJAX</title>
<script type="text/javascript">

    var http = getXMLHttpRequest();

/* Crea un'istanza dell'oggetto XMLHttpRequest come risultato.
In diverse versioni dei browser la generazione di questo oggetto
richiede l'attivazione delle classi predefinite */
    function getXMLHttpRequest() {
        try {
            req = new XMLHttpRequest();
        } catch (err1) {
            try {
                req = new ActiveXObject("Msxml2.XMLHTTP");
            } catch (err2) {
                try {
                    req = new ActiveXObject("Microsoft.XMLHTTP");
                } catch (err3) {
                    req = false;
                }
            }
        }
        return req;
    }

    function getProvincia(regione) {
/* Se la regione non è stata selezionata, azzera le opzioni
presenti all'interno della selezione della provincia */
        if (regione == "-- Seleziona --") {
            document.forms['miaform'].provincia.options.length = 0;
            return;
        }
        var modurl = 'jq11.php?reg=' + regione;

        // Inizializza XMLHttpRequest - il terzo parametro indica
        http.open("GET", modurl, true);
    }

```

```

        // Definisce la funzione di callback
        http.onreadystatechange = useHttpResponse;

        // Invia la richiesta al server
        http.send();
    }

    function useHttpResponse() {
        /*
            // Valori di readyState che rappresentano

            0: Uninitialized. Oggetto non inizializzato
            1: Open. Attivato con successo im metodo
            2: Sent. Richiesta inoltrata al server
            3: Receiving. Ricevuto l'header della ris
            4: Loaded. La risposta stata ricevuta
        */
        if (http.readyState == 4) {
            if (http.status == 200) {
                // Risposta OK
                var provList = http.responseXML.ge
                document.forms['miaform'].provinci
                for (i = 0; i < provList.length; i
                    document.forms['miaform'].prov
                }
            }
        }
    }
}

/* All'atto della terminazione del caricamento della pagina
definita una funzione che seleziona all'interno del DOM l'
ha id pari a selReg e assegna all'evento onchange l'handle
dalla successiva funzione */
window.onload = function () {
    document.getElementById('selReg').onchange
/* Viene attivata la funzione getProvincia() che si vede p
il contenuto dell'opzione selezionata, da cui estraiamo la
associata a quella opzione*/

```

```

        getProvincia(this.options[this.selectedIndex].text);
    };
};

</script>
</head>

<body>
    <h1>Applicazione Ajax su Select</h1>
    <form name="miaform" action="">
        <label>Regione
            <select id="selReg" name="regione" size="1">
                <option selected>-- Seleziona --</option>
                <option>Marche</option>
                <option>Abruzzo</option>
            </select><br>
        </label>
        <label>Provincia
            <select name="provincia" size="1">
            </select>
        </label>
    </form>
</body>
</html>

```

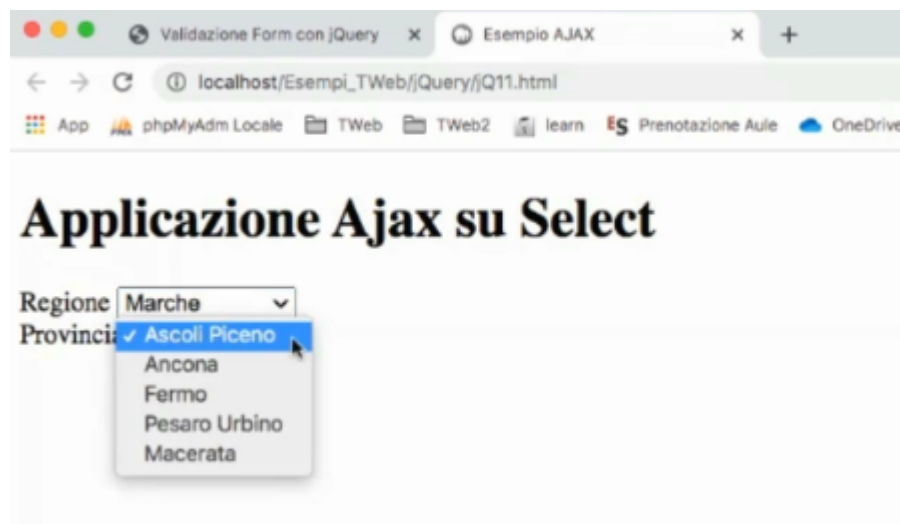
```

<?php

$parm = $_GET['reg'];
$conn = mysqli_connect('localhost', 'lpweb', 'lpweb', 'Ajax');
$query = "select Prov from Province where Reg='" . $parm . "'";
$ris = mysqli_query($conn, $query);
header('Content-Type: text/xml');
echo '<?xml version="1.0"?>';
echo '<oggetto>';
while ($row = mysqli_fetch_assoc($ris)) {
    echo '<provincia>' . $row['Prov'] . '</provincia>';
}
echo '</oggetto>';

```

```
mysqli_free_result($ris);  
mysqli_close($conn);
```



Interazione AJAX con jQuery

| ajax() – Utility Function | |
|-------------------------------|--|
| <code>\$.ajax(options)</code> | Invia una richiesta AJAX secondo le modalità specificate dall'oggetto <code>options</code> . |
| options (set non completo) | |
| <code>url</code> | L'url del programma sul server che riceve la richiesta |
| <code>type</code> | Il metodo HTTP (GET o POST) da usare per l'inoltro della richiesta. |
| <code>data</code> | I dati inviati nella richiesta al server (Types: String Object Array) |
| <code>dataType</code> | Codifica attesa dei dati ricevuti dal server <code>xml html json text ...</code> |
| <code>success</code> | Nome della funzione di callback |

Esempio jQuery#12

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Esempio AJAX</title>
  // Scarico la libreria jQuery da questo URL
  <script src="https://code.jquery.com/jquery-3.3.1.min"
  <script type="text/javascript">
    $(function(){
      $('#selReg').change(function(event){
        $.ajax({
          type: 'GET',
          url: 'jq11.php',
          data: "reg=" + $('#selReg').val(),
          dataType: 'xml',
          success: setProvince
        });
      });
    });

    function setProvince(xml){
      $('#selProv').find('option').remove();
      $(xml).find('provincia').each(function(){
```

```

/* Aggiunge attraverso il metodo append la stringa associata
elemento al blocco delle option della select che elencherà le
della nostra regione */
        $('#selProv').append('<option>' + $(this)
    });
    }
</script>
</head>
<body>
    <h1>Applicazione Ajax su Select</h1>
    <form name="miaform" action="">
        <label>Regione <select id="selReg" name="regione"
            <option selected>-- Seleziona --</opt
            <option>Marche</option>
            <option>Abruzzo</option>
        </select><br>
        </label>
        <label>Provincia <select id="selProv" name="provi
    </form>
</body>
</html>

```

Codifica JSON

JSON (*JavaScript Object Notation*) : notazione JS che consente di rappresentare in maniera semplice un oggetto per mezzo di una stringa che contiene tra parentesi graffe le coppie `nome:valore` della proprietà dell'oggetto, separate da virgola.

Notazione ricorsiva : se un oggetto ne contiene un altro, quest'ultimo viene rappresentato allo stesso modo.

```
var proprietario = new Object();
proprietario.nome = 'Mario';
proprietario.cognome = 'Rossi';

var moto = new Object();
moto.modello = 'Ducati Diavel';
moto.anno = 2011;
moto.telaio = 156298876;
moto.proprietario = proprietario;
```

Notazione classica



```
var moto = {
  modello: 'Ducati Diavel',
  anno: 2011,
  telaio: 156298876,
  proprietario: {
    nome: 'Mario',
    cognome: 'Rossi'
  }
}
```

JSON

AJAX JSON

Il formato JSON risulta più efficiente nello scambio di dati tra client e server in una sessione AJAX perchè :

- la codifica dei dati è più sintetica
- il parser JSON lato client è più efficiente

JSON è diventato lo standard di fatto nelle interazioni AJAX.

Variante al formato standard : sia i nomi che i valori delle componenti degli oggetti sono racchiusi tra apici doppi (").

Esempio jQuery#13

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Esempio AJAX</title>
    <script src="https://code.jquery.com/jquery-3.3.1.min">
    <script type="text/javascript">
      $(function () {
        $('#selReg').change(function (event) {
          if ($('#selReg').val() == "-- Seleziona -"
            $('#selProv').find('option').remove()
          return
        })
        $.ajax({
          type: 'GET',
```

```

        url: 'jq13.php',
        data: "reg=" + $('#selReg').val(),
        dataType: 'json',
        success: setProvince
    });
});
});

function setProvince(data) {
    $('#selProv').find('option').remove();
    $.each(data, function (key, val) {
        $('#selProv').append('<option>' + val + ' ');
    });
}
</script>
</head>
<body>
    <h1>Applicazione Ajax su Select</h1>
    <form name="miaform" action="">
        <label>Regione <select id="selReg" name="regione"
            <option selected>-- Seleziona --</option>
            <option>Marche</option>
            <option>Abruzzo</option>
        </select><br>
        </label>
        <label>Provincia
            <select id="selProv" name="provincia" size="1"
            </select>
        </label>
    </form>
</body>
</html>

```

```
<?php
```

```

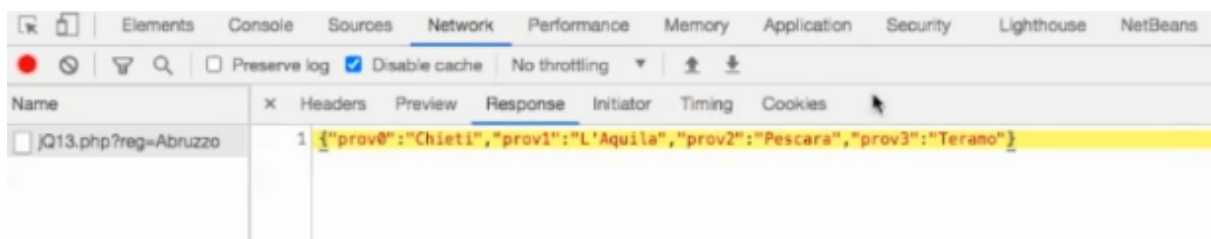
$parm = $_GET['reg'];
$conn = mysqli_connect('localhost', 'lpweb', 'lpweb', 'Ajax')
$query = "select Prov from Province where Reg='" . $parm . "'

```

```

$ris = mysqli_query($conn, $query);
$i = 0;
foreach ($ris as $prov) {
    $data[$i++] = $prov['Prov'];
};
header('Content-Type: application/json');
// Passiamo il risultato di una query db e la trasforma in JS
echo json_encode($data);
mysqli_free_result($ris);
mysqli_close($conn);

```



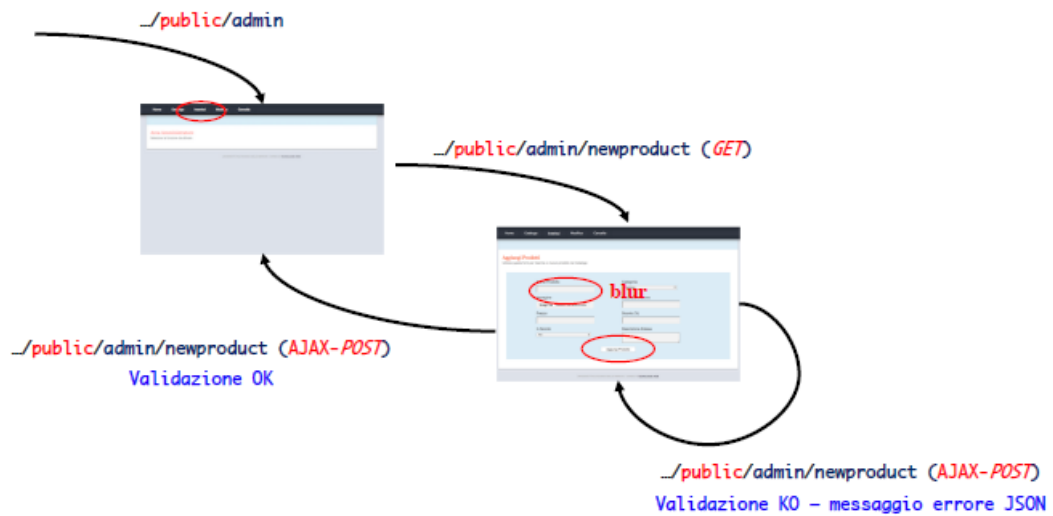
▼ Esempio laraProj6

- Estensione di laraProj5
- Validazione incrementale valori FORM inserimento prodotti lato server

Obiettivi didattici : illustrare

- Uso AJAX-JSON via jQuery integrato in Laravel

laraProj6: routing



▼ admin.blade.php

File contenuto all'interno della cartella *views/layouts*

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale())"
  <head>
    <meta charset="utf-8">

    /* Inseriamo la sezione link per creare un nuovo schema
    la possibilità alle liste non solo di iniettare nella s
    gli script che serviranno per la realizzazione delle si
    ma anche di poter iniettare dei nuovi link (come in que
    @section('link')
    <link rel="stylesheet" type="text/css" href="{{
    @show
    @section('scripts')
    @show
    <title>LaProj6 | @yield('title', 'Catalogo')</t
  </head>
  <body id="bodyadmin">
    <div id="wrapper">
      <div id="menu">
        @include('layouts/_navadmin')
```

```

        </div>

        <!-- end #menu -->
        <div id="page">
            <div id="page-bgtop">
                <div id="page-bgbtm">
                    @yield('content')
                    <div style="clear: both;">&nbsp;
                </div>
            </div>
        </div>

        <!-- end #content -->
        <div id="footer">
            <br>
            <p>universit&agrave; politecnica delle
        </div>
        <!-- end #footer -->
    </div>
</body>
</html>

```

▼ insert.blade.php

In questa versione il processo di segnalazione degli errori avviene per iniezione dei messaggi di errore all'interno della stessa pagina nella quale l'utente ha inserito il dato errato (e non attraverso il caricamento di una nuova pagina come avveniva precedentemente).

```

@extends('layouts.admin')

@section('title', 'Area Admin')

@section('scripts')
/* Direttiva utilizzata per includere contenuti di tipo
definiti a livello di layout da cui la vista eredita (n
caso non ci sono) */
@parent
/* All'interno di functions.js andremo a codificare tut

```

```

funzionalità JS che implementano il meccanismo di validazione
interattivo da poter utilizzare anche in altre form*/
<script src="{ asset('js/functions.js') }" ></script>
// Carichiamo la libreria jQuery dal sito delle Google
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js">
// Codice che costituisce lo scheletro della validazione
<script>
$(function () {
// Azione da validare
    var actionUrl = "{ route('newproduct.store') }";
// id della form
    var formId = 'addproduct';
/* Event handler (associato a tutti gli elementi di input
associato all'evento blur , quindi alla perdita del focus
qualunque di questi elementi (singolo elemento)*/
    $(":input").on('blur', function (event) {
/* Al verificarsi di tale evento attiveremo una funzione che
valida l'elemento della form su cui si è verificato il blur ed
associa l'azione da validare */
        var formElementId = $(this).attr('id');
/* Questa funzione è definita all'interno di functions.js
e contiene i contenuti di un singolo elemento */
        doElemValidation(formElementId, actionUrl, formId);
    });
/* Event handler associato all'evento submit (associato
alla pagina identificato dall'input #addproduct (la nostra
funzione di validazione) */
    $("#addproduct").on('submit', function (event) {
/* Interrompe il processo predefinito di submit della form
il processo verrà gestito dalla funzione presente sulla riga 10
*/
        event.preventDefault();
// Valida tutti i contenuti dell'intera form
        doFormValidation(actionUrl, formId);
    });
});
</script>

@endsection

@section('content')

```



```

<div class="static">
  <h3>Aggiungi Prodotti</h3>
  <p>Utilizza questa form per inserire un nuovo prodo

  <div class="container-contact">
    <div class="wrap-contact">
      {{ Form::open(array('route' => 'newproduct.
      <div class="wrap-input rs1-wrap-input">
        {{ Form::label('name', 'Nome Prodotto',
        {{ Form::text('name', '', ['class' => '
      </div>

      <div class="wrap-input rs1-wrap-input">
        {{ Form::label('catId', 'Categoria', ['
        {{ Form::select('catId', $cats, '', ['c
      </div>

      <div class="wrap-input rs1-wrap-input">
        {{ Form::label('image', 'Immagine', ['c
        {{ Form::file('image', ['class' => 'inp
      </div>

      <div class="wrap-input rs1-wrap-input">
        {{ Form::label('descShort', 'Descrizione
        {{ Form::text('descShort', '', ['class'
      </div>

      <div class="wrap-input rs1-wrap-input">
        {{ Form::label('price', 'Prezzo', ['cla
        {{ Form::text('price', '', ['class' =>
      </div>

      <div class="wrap-input rs1-wrap-input">
        {{ Form::label('discountPerc', 'Sconto
        {{ Form::text('discountPerc', '', ['cla
      </div>

      <div class="wrap-input rs1-wrap-input">

```

```

        {{ Form::label('discounted', 'In Sconto') }}
        {{ Form::select('discounted', ['1' => '1'], $discounted) }}
    </div>

    <div class="wrap-input rs1-wrap-input">
        {{ Form::label('descLong', 'Descrizione') }}
        {{ Form::textarea('descLong', '', ['class' => 'form-control']) }}
    </div>

    <div class="container-form-btn">
        {{ Form::submit('Aggiungi Prodotto', ['class' => 'btn btn-primary']) }}
    </div>

    {{ Form::close() }}
</div>
</div>
@endsection

```

▼ functions.js

```

function getErrorHtml(elemErrors) {
    if ((typeof (elemErrors) === 'undefined') || (elemErrors === '')) {
        return;
    }
    var out = '<ul class="errors">';
    for (var i = 0; i < elemErrors.length; i++) {
        out += '<li>' + elemErrors[i] + '</li>';
    }
    out += '</ul>';
    return out;
}

function doElemValidation(id, actionUrl, formId) {

    var formElems;

    function addFormToken() {

```

```

        var tokenVal = $("#" + formId + " input[name=_token]");
        formElems.append('_token', tokenVal);
    }

    function sendAjaxReq() {
        $.ajax({
            type: 'POST',
            url: actionUrl,
            data: formElems,
            dataType: "json",
            error: function (data) {
                if (data.status === 422) {
                    var errMsgs = JSON.parse(data.responseText);
                    $("#" + id).parent().find('.errors').html('');
                    $("#" + id).after(getErrorHtml(errMsgs));
                }
            },
            contentType: false,
            processData: false
        });
    }

    var elem = $("#" + formId + " :input[name=" + id + "]");
    if (elem.attr('type') === 'file') {
        // elemento di input type=file valorizzato
        if (elem.val() !== '') {
            inputVal = elem.get(0).files[0];
        } else {
            inputVal = new File([""], "");
        }
    } else {
        // elemento di input type != file
        inputVal = elem.val();
    }

    formElems = new FormData();
    formElems.append(id, inputVal);
    addFormToken();
    sendAjaxReq();

```

```

}

function doFormValidation(actionUrl, formId) {

    var form = new FormData(document.getElementById(formId));
    $.ajax({
        type: 'POST',
        url: actionUrl,
        data: form,
        dataType: "json",
        error: function (data) {
            if (data.status === 422) {
                var errMsgs = JSON.parse(data.responseText);
                $.each(errMsgs, function (id) {
                    $("#"+id).parent().find('.errors').html('');
                    $("#"+id).after(getErrorHtml(errMsgs[id]));
                });
            }
        },
        success: function (data) {
            window.location.replace(data.redirect);
        },
        contentType: false,
        processData: false
    });
}

```

▼ NewProductRequest.php

```

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Validation\Rule;

// Aggiunti per response JSON

```

```

use Illuminate\Http\Exceptions\HttpResponseException;
use Illuminate\Contracts\Validation\Validator;
use Symfony\Component\HttpFoundation\Response;

class NewProductRequest extends FormRequest {

    /**
     * Determine if the user is authorized to make this
     *
     * @return bool
     */
    public function authorize() {
        // Nella form non mettiamo restrizioni d'uso su
        // Gestiamo l'autorizzazione ad un altro livello
        return true;
    }

    /**
     * Get the validation rules that apply to the request
     *
     * @return array
     */
    public function rules() {
        return [
            'name' => 'required|max:25',
            'catId' => 'required',
            'descShort' => 'required|max:30',
            'image' => 'file|mimes:jpeg,png|max:1024',
            'price' => 'required|numeric|min:0',
            'discountPerc' => 'required|integer|min:0|max:100',
            'discounted' => 'required',
            'descLong' => 'required|max:2500'
        ];
    }

    /**
     * Override: response in formato JSON
     */
}

```

```

        protected function failedValidation(Validator $validator)
        {
            throw new HttpResponseException(response($validator->errors));
        }
    }
}

```

▼ AdminController.php

```

<?php

namespace App\Http\Controllers;

use App\Models\Admin;
use App\Models\Resources\Product;
use App\Http\Requests\NewProductRequest;

class AdminController extends Controller {

    protected $_adminModel;

    public function __construct() {
        $this->middleware('can:isAdmin');
        $this->_adminModel = new Admin;
    }

    public function index() {
        return view('admin');
    }

    public function addProduct() {
        $prodCats = $this->_adminModel->getProdsCats();
        return view('product.insert')
            ->with('cats', $prodCats);
    }

    public function storeProduct(NewProductRequest $req

```

```

        if ($request->hasFile('image')) {
            $image = $request->file('image');
            $imageName = $image->getClientOriginalName(
        } else {
            $imageName = NULL;
        }

        $product = new Product;
        $product->fill($request->validated());
        $product->image = $imageName;
        $product->save();

        if (!is_null($imageName)) {
            $destinationPath = public_path() . '/images';
            $image->move($destinationPath, $imageName);
        };

        return response()->json(['redirect' => route('a
    }

}

```