



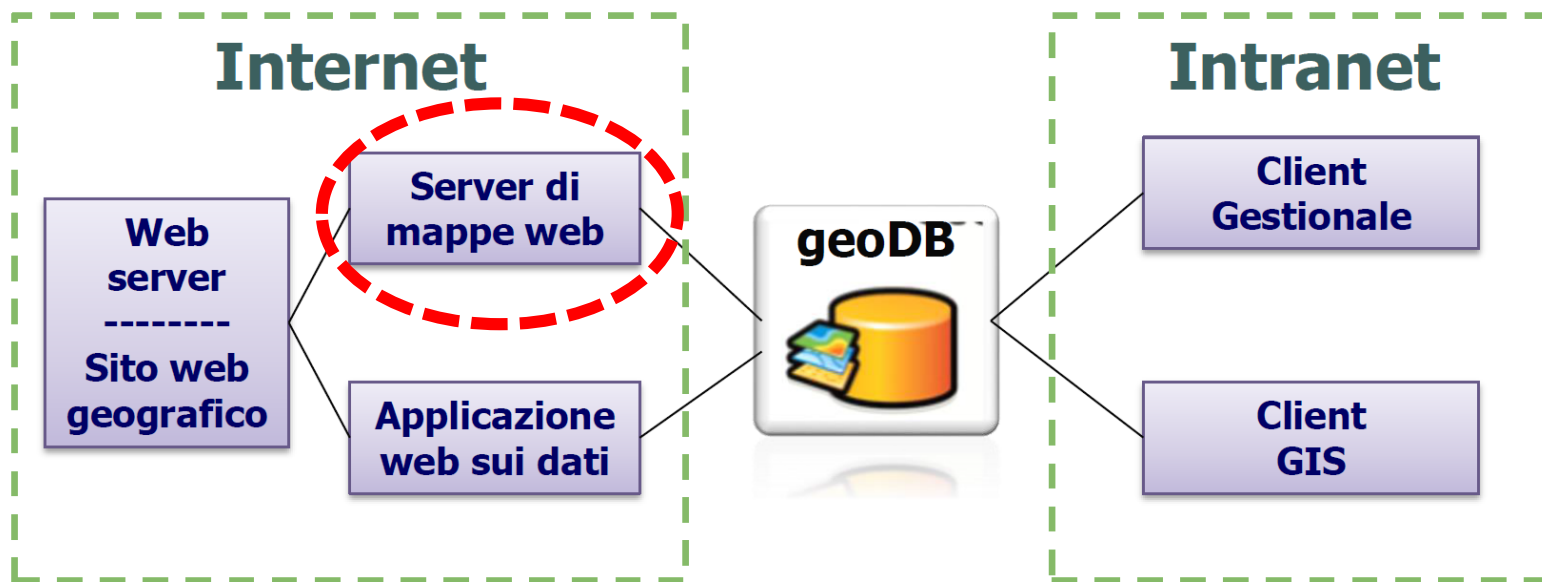
Strumenti e tecnologie Geo-web

Fondamenti sulla condivisione web delle informazioni geografiche

Corso di Sistemi Informativi Territoriali Avanzati – UD12

prof. Giovanni Borga

Architettura di sistema per una piattaforma geo-web



Interoperabilità e Open Geospatial Consortium

About OGC

The OGC (Open Geospatial Consortium) is an international not for profit organization committed to making **quality open standards for the global geospatial community**. These standards are made through a consensus process and are freely available for anyone to use to improve sharing of the world's geospatial data.

OGC members come from government, commercial organizations, NGOs, academic and research organizations.

Strategic (5)	Technical Aggregate (1)	GovFuture-Subnational (20)	University (110)
Principal (18)	Associate (127)	NGO / Not For Profit Institute (57)	Individual (32)
Technical (74)	Small Company (44)	GovFuture-Local (33)	

Standards di interoperabilità OGC

Standard attualmente più utilizzati:

WMS

WFS

www.opengeospatial.org



About Standards Innovation News & Events Membership Resources

OGC Standards

Below is a list of OGC Implementation Standards.

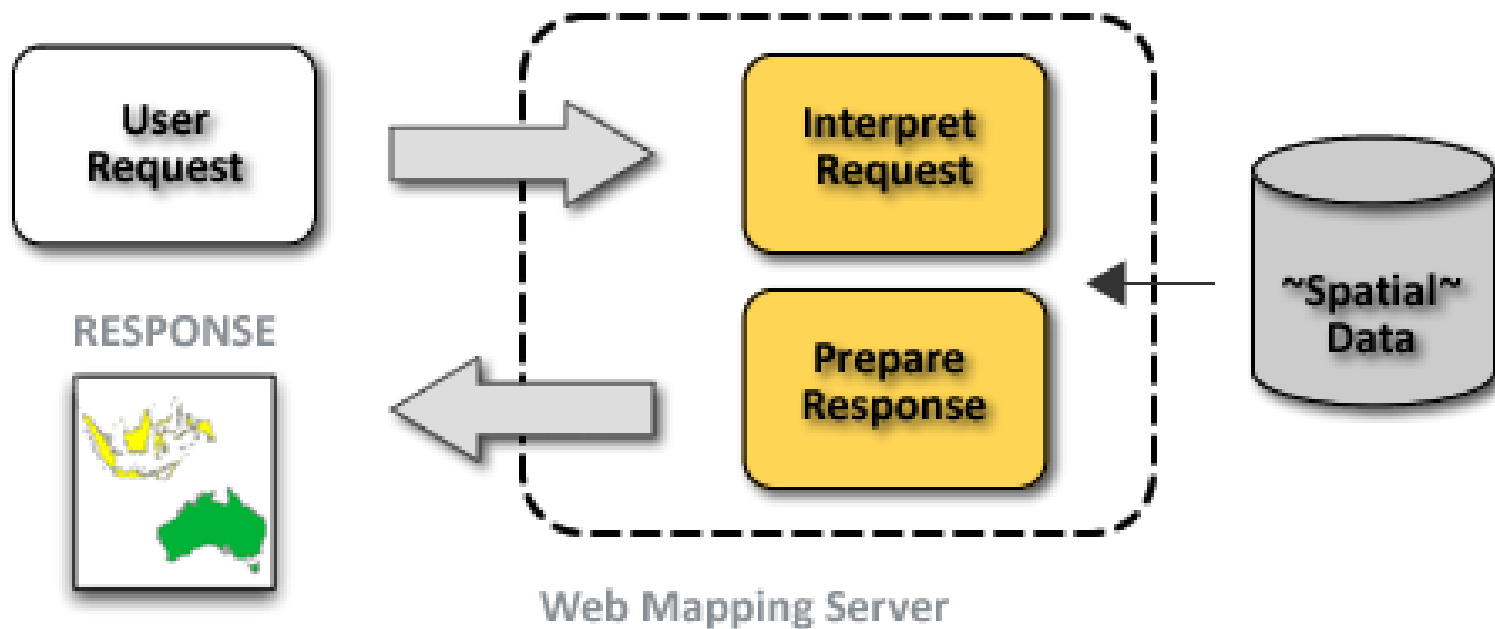
Implementation Standards are different from the Abstract Specification. They are written for a more technical audience and detail the interface structure between software components. An interface specification is considered to be at the implementation level of detail if, when implemented by two different software engineers in ignorance of each other, the resulting components plug and play with each other at that interface.

Any Schemas (xsd, xsilt, etc) that support an approved Implementation Standard can be found in the official [OGC Schema Repository](#).

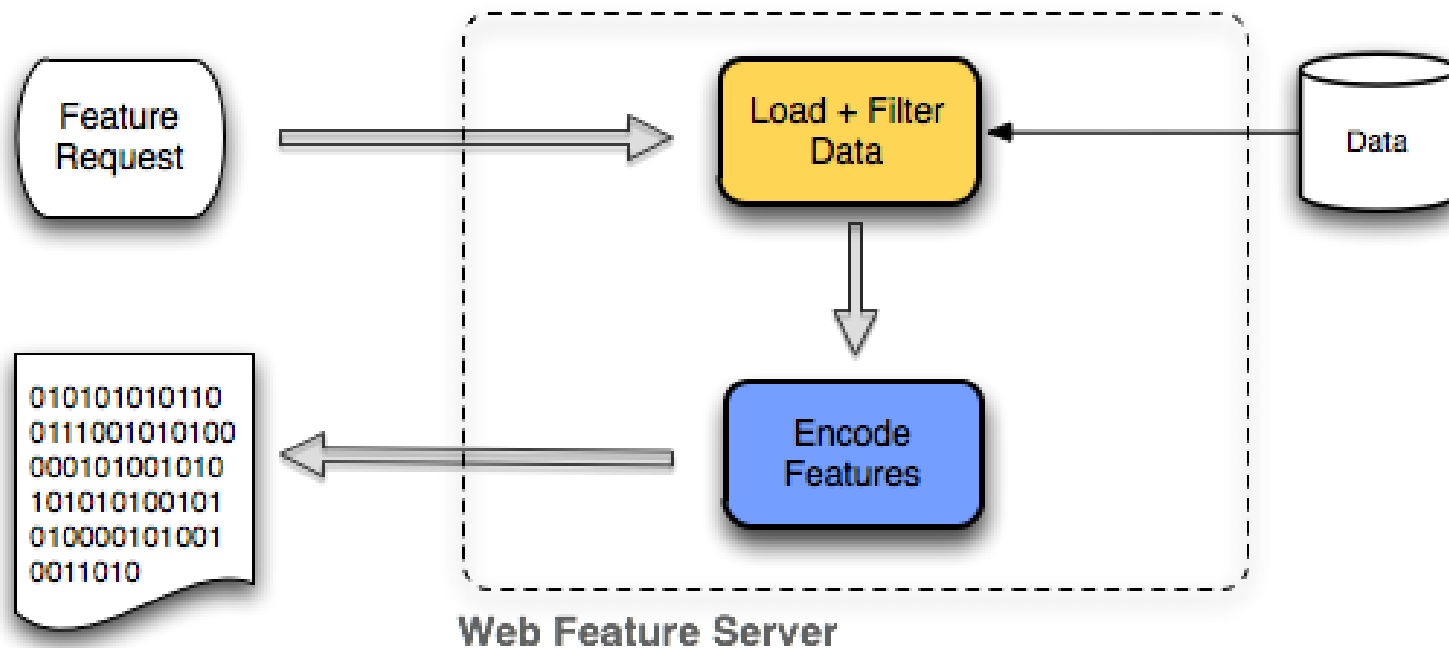
Document	Title (click to view/download)	Version	Doc.#	Editor	Date
	CF-netCDF3 Data Model Extension standard netcdf-data-model-ex	3.1	11-165r2	Ben Domenico and Stefano Nativi	2013-01-0
<p>The OGC netCDF encoding supports electronic encoding of geospatial data, that is, digital geospatial information representing space and time-varying phenomena. This standard specifies the CF-netCDF data model extension. This standard specifies the CF-netCDF data model mapping onto the ISO 19123 coverage schema. This standard deals with multi-dimensional gridded data and multi-dimensional multi-point data. In particular, this extension standard encoding profile is limited to multi-point, and regular and warped grids; however, irregular grids are important in the CF-netCDF community and work is underway to expand the CF-netCDF to encompass other coverages types, including irregular gridded datasets.</p> <p>See more...</p>					
	Corrigendum 1 for OGC Web Services Common Standard v2.0.0 - Multilingual CommonC1		11-157	Jim Greenwood	2011-10-1
<p>This document being corrected specifies many of the aspects that are, or should be, common to all or multiple OWS interface Implementation Specifications. The Common Implementation Specification aspects specified by this document currently include: a) Operation request and response contents, most partial b) Parameters and data structures included in operation requests and responses c) XML and KVP encoding of operation requests and responses</p> <p>See more...</p>					
	Corrigendum 2 for OGC Web Services Common Specification v 1.1.0 - Exception Report CommonC2		11-158	Jim Greenwood	2011-10-18
<p>This document defines the corrigendum change notes for See more...</p>					
	Corrigendum for OpenGIS Implementation Standard Web Processing Service (WPS) 1.0.0 WPS 1.0.0 Cor	0.0.8	08-0916	Peter Schut	2009-09-16
<p>This document provides the details for a corrigendum for the existing OpenGIS Standard for the Web Processing Service version 1.0.0 and does not modify that standard. The current OpenGIS Implementation Standard that this document provides revision notes for is: 05-007r7.</p> <p>See more...</p>					
	CSW-ebRIM Registry Service - Part 1: ebRIM profile of	1.0.1	07-110r4	Richard Martell	2009-02-05

- OGC Standards
 - ARML2.0
 - Cat: ebRIM App Profile: Earth Observation Products
 - Catalogue Service
 - CityGML
 - Coordinate Transformation
 - Filter Encoding
 - GML in JPEG 2000
 - GeoAPI
 - GeoPackage
 - GeoSparql
 - Geography Markup Language
 - Geospatial eXtensible Access Control Markup Language (GeoXACML)
 - IndoorGML
 - KML
 - Location Services (OpenLS)
 - Moving Features
 - NetCDF
 - Observations and Measurements
 - Open GeoSMS
 - OpenMI
 - OpenSearch Geo
 - Ordering Services Framework for Earth Observation Products
 - OWS Context
 - PUCK
 - SWE Common Data Model
 - SWE Service Model
 - Sensor Model Language
 - Sensor Observation Service
 - Sensor Planning Service
 - Simple Features
 - Simple Features CORBA
 - Simple Features OLE/COM
 - Simple Features SQL
 - Styled Layer Descriptor
 - Symbolization Encoding

Web Map Service

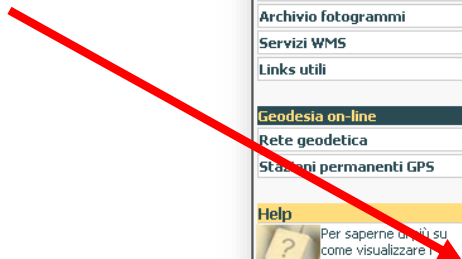


Web Feature Service



Esempio di WMS pubblico: regione Abruzzo

Link al servizio



Sei in: Regione Abruzzo | Cartografia on-line | Servizi WMS

Servizi
Consultazione e distribuzione
Richiesta
Listino prezzi

Cartografia on-line
Carte on-line
Archivio fotogrammi
Servizi WMS
Links utili

Geodesia on-line
Rete geodetica
Stazioni permanenti GPS

Help
Per saperne di più su come visualizzare i fotogrammi e la cartografia on-line, come richiedere copie delle foto aeree.

Web Map Services
WMS Web Map Service - ISO19128 Web Map Server Interface
È un protocollo standard OGC, tramite il quale è possibile generare dinamicamente mappe di dati spazialmente riferiti.
In particolare lo standard ISO19128 specifica il comportamento di un servizio web che produce dinamicamente mappe di dati spazialmente riferiti a partire da informazioni geografiche, sottoforma di immagini. Questo standard internazionale definisce una "mappa" come rappresentazione di informazioni geografiche, restituendo un'immagine digitale idonea ad essere visualizzata sullo schermo di un computer. Dal 2005 la specifica "OGC WMS" è divenuta uno standard ISO ed identificata con il codice ISO19128 - Web map server interface.
Il server raster.regione.abruzzo.it WMS, basato su tecnologia ERDAS IMAGE MANAGER provvede alla fornitura di layer geografici tramite protocollo di richiesta WMS. Le immagini fornite vengono create sulla base di dati geografici in formato raster: ecw, tiff, jpeg2000 etc...

La lista dei servizi attualmente forniti è disponibile in formato XML nel seguente url:

- http://raster.regione.abruzzo.it/ecwp/ecw_wms.dll?request=GetCapabilities&service=wms

Dal seguente link è possibile richiedere i dati dei servizi in formato tabellare:

- Lista dei servizi

Diagramma: ERDAS Image Manager (contiene ECWP Delivery, Ortho-Color Balance & Mosaic, OGC WMS Client (ECW)) è collegato a OGC Web Client (RedSpider Enterprise), che a sua volta è collegato a GIS/CAD Systems e ERDAS Data Client (ECW). Il diagramma mostra anche database (Oracle, Microsoft SQL Server, IBM DB2) e un server (Securely Discover, Describe).

Progetti
Sistema SigmaTer
Salix Herbacea
Rete Geodetica
Vincolo idrogeologico
ComNet-RA
DB Topografico
Carta Tecnica Regionale
Misura GPS della quota del Gran Sasso

Documentazione
Normativa
Attività
Laboratorio
Pubblicazioni

Archivio
Cartografia a stampa
Cartografia numerica
Cartografia storica
Modello digitale terreno
Foto aeree

Anatomia del servizio web WMS

http://raster.regione.abruzzo.it/ecwp/ecw_wms.dll?request=GetCapabilities&service=wms

- HTTP-based
- Combine map images from multiple remote servers
- Request types:
 - **GetCapabilities**
 - GetMap
 - GetFeatureInfo

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- ER Mapper Image Web Server 8.5 -->
<!DOCTYPE WMT_MS_Capabilities (View Source for full doctype...)>
- <WMT_MS_Capabilities version="1.1.1" updateSequence="0">
+ <Service>
- <Capability>
+ <Request>
+ <Exception>
+ <Layer queryable="0" opaque="0" noSubsets="0">
</Capability>
</WMT_MS_Capabilities>
```

Le sotto-sezioni del GetCapabilities:

- Request (tipi di richieste supportate; es. GetMap)
- Exception (tipi di dato per i messaggi di errore)
- Layer (layers disponibili e loro caratteristiche)

Anatomia del servizio web WMS

La sotto-sezione Layer del
GetCapabilities ...

```
<Layer>
  <Title>Root Layer Name</Title>
  ...

  <Layer>
    <Title>Groupe/Theme Name</Title>
    ...

    <Layer>
      <Title>Layer 1</Title>
      ...
    </Layer>

    <Layer>
      <Title>Layer 2</Title>
      ...
    </Layer>

    <Layer>
      <Title>Layer 3</Title>
      ...
    </Layer>

</Layer>
```

Anatomia del servizio web WMS

http://raster.regione.abruzzo.it/ecwp/ecw_wms.dll?request=GetCapabilities&service=wms

- HTTP-based
- Combine map images from multiple remote servers
- Request types:
 - **GetCapabilities**
 - GetMap
 - GetFeatureInfo

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- ER Mapper Image Web Server 8.5 -->
<!DOCTYPE WMT_MS_Capabilities (View Source for full doctype...)>
- <WMT_MS_Capabilities version="1.1.1" updateSequence="0">
- <Service>
  <Name>OGC:WMS</Name>
  <Title>Abruzzo servizi WMS Raster</Title>
  <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:
    xlink:href="http://raster.regione.abruzzo.it" />
  <Abstract>WMS Regione Abruzzo</Abstract>
- <KeywordList>
  <Keyword>ECW</Keyword>
  <Keyword>JPEG 2000</Keyword>
  <Keyword>Abruzzo</Keyword>
  <Keyword>cartografia</Keyword>
  <Keyword>raster</Keyword>
  <Keyword>wms</Keyword>
  <Keyword>regione</Keyword>
  <Keyword>servizi</Keyword>
  <Keyword>web map service</Keyword>
  </KeywordList>
- <ContactInformation>
- <ContactPersonPrimary>
  <ContactPerson>Alessandro Cacchione</ContactPerson>
  <ContactOrganization>Regione Abruzzo</ContactOrganization>
  </ContactPersonPrimary>
  <ContactPosition>Webgis area services manager</ContactPosition>
- <ContactAddress>
  <AddressType>indirizzo</AddressType>
  <Address>via Leonardo Da Vinci 6</Address>
  <City>L'Aquila</City>
  <StateOrProvince>Abruzzo-Italy</StateOrProvince>
  <PostCode>67100</PostCode>
```

Anatomia del servizio web WMS

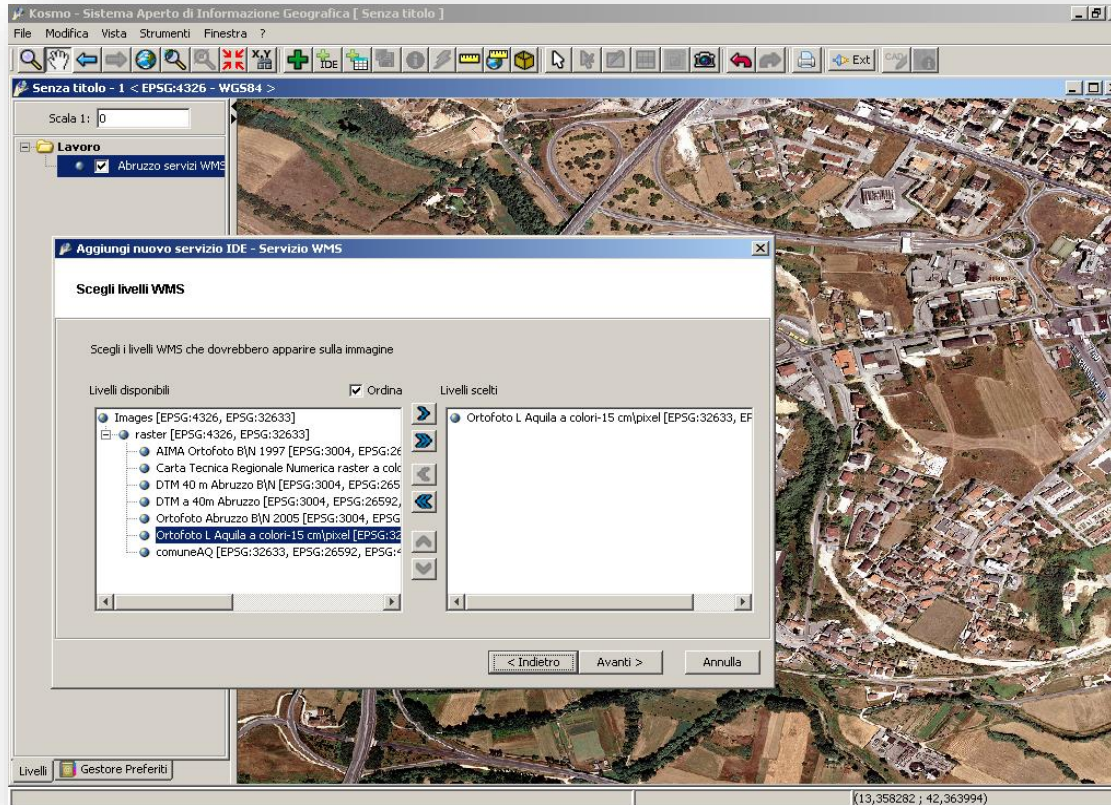
http://raster.regione.abruzzo.it/ecwp/ecw_wms.dll

```
?request=GetMap
&version=1.1.0
&layers=IMAGES_COMUNEAQ.ECW
&styles=
&srs=EPSG:4326
&bbox=13.36,42.36,13.37,42.35
&width=400&height=350&format=image/jpeg
&transparent=false
```



- HTTP-based
- Combine map images from multiple remote servers
- Request types:
 - GetCapabilities
 - **GetMap**
 - GetFeatureInfo

Accesso ad un WMS tramite un desktop GIS





Geoserver

Geoserver

What is Geoserver?

GeoServer is a Java-based software server that allows users to view and edit geospatial data. Using open standards set forth by the [Open Geospatial Consortium \(OGC\)](#), GeoServer allows for great flexibility in map creation and data sharing.

Open and Share Your Spatial Data

GeoServer allows you to display your spatial information to the world. Implementing the [Web Map Service \(WMS\)](#) standard, GeoServer can create maps in a variety of output formats. [OpenLayers](#), a free mapping library, is integrated into GeoServer, making map generation quick and easy. GeoServer is built on [Geotools](#), an open source Java GIS toolkit.

There is much more to GeoServer than nicely styled maps, though. GeoServer also conforms to the [Web Feature Service \(WFS\)](#) standard, which permits the actual sharing and editing of the data that is used to generate the maps. Others can incorporate your data into their websites and applications, freeing your data and permitting greater transparency.

Use Free and Open Source Software

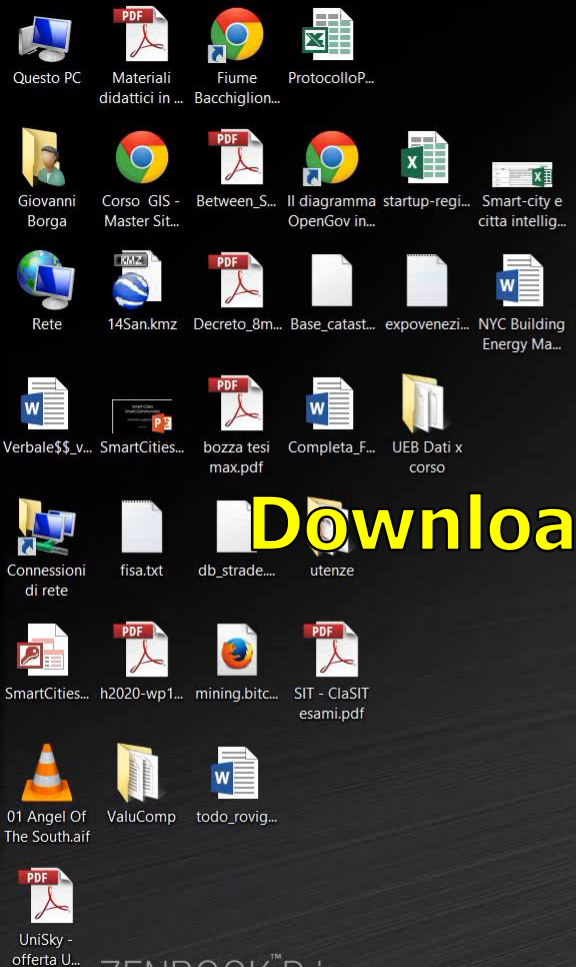
GeoServer is [free software](#). This significantly lowers the financial barrier to entry when compared to traditional GIS products. In addition, not only is it available free of charge, it is also open source. Bug fixes and feature improvements in open source software are greatly accelerated when compared to traditional software solutions. Leveraging GeoServer in your organization also prevents software lock-in, saving costly support contracts down the road.

Integrate With Existing Mapping APIs

GeoServer can display data on any of the popular mapping applications such as [Google Maps](#), [Google Earth](#), [Yahoo Maps](#), and [Microsoft Virtual Earth](#). In addition, GeoServer can connect with traditional GIS architectures such as [ESRI ArcGIS](#).

Join the Community

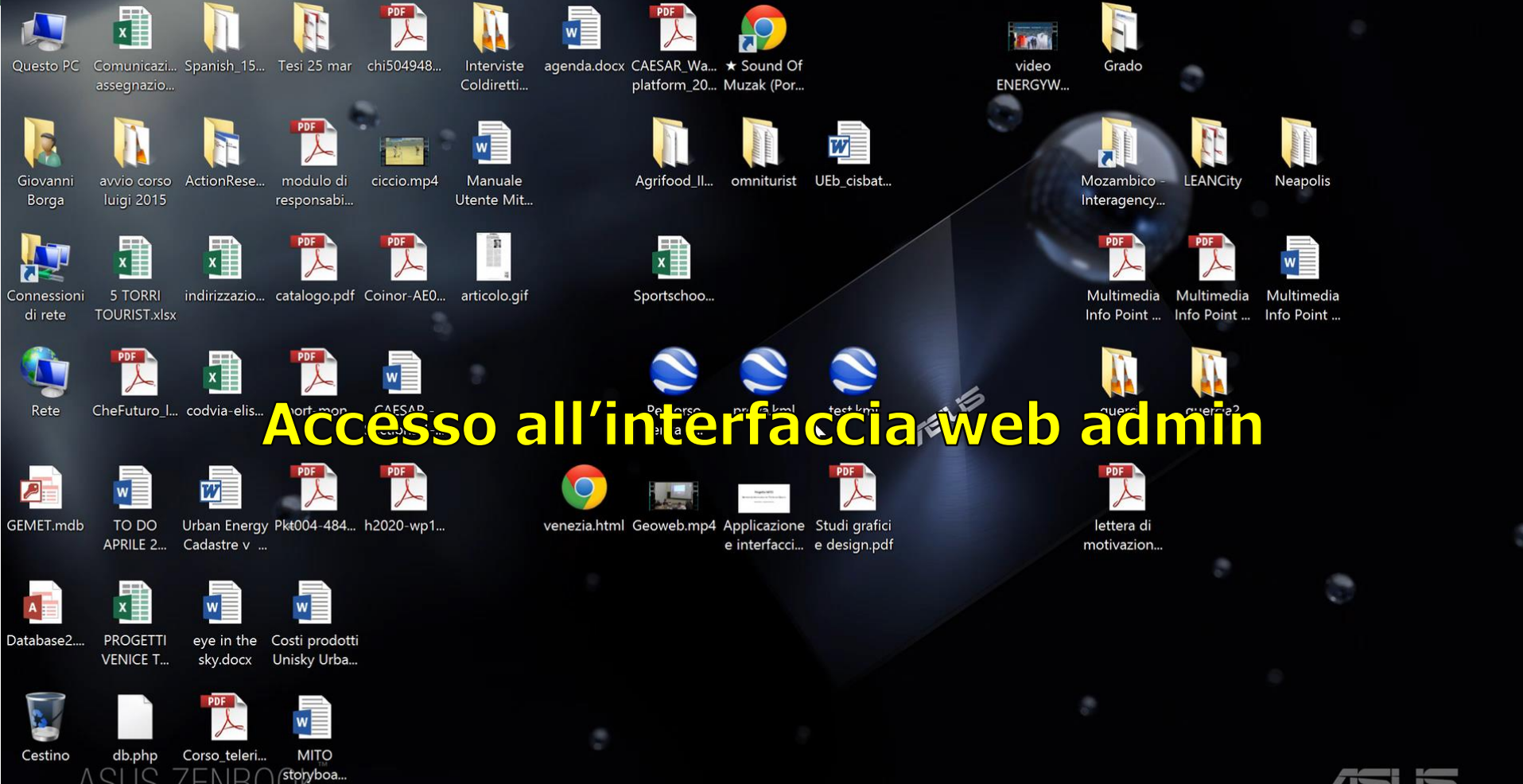
GeoServer has a large and vibrant community consisting of users and developers from all over the world. Support is available through a variety of sources, such as [email lists](#) and [IRC](#) (the latter with real time access to software developers). In addition, [commercial support](#) is available from a variety of providers. With GeoServer, you are always in good company.



Download e installazione di geoserver



Corso di Laurea Magistrale in Pianificazione e Politiche per la Città, il Territorio e l'Ambiente



Accesso all'interfaccia web admin



Username

Password

Remember me

Login

Welcome

Welcome

This GeoServer belongs to [The ancient geographies INC.](#)

This GeoServer instance is running version **2.7.1**. For more information please contact the administrator.

Service Capabilities

WCS

1.0.0

1.1.0

1.1.1

1.1

1.0

1.0.0

1.1.0

2.0.0

WMS

1.1.1

1.3.0

TMS

1.0.0

WMS-C

1.1.1

WMTS

1.0.0

Creazione di un workspace

About & Status

[About GeoServer](#)

Data

[Layer Preview](#)

Demos



Logged in as admin.

[Logout](#)

Workspaces

Manage GeoServer workspaces

- [Add new workspace](#)
- [Remove selected workspace\(s\)](#)

<< < 1 > >> Results 1 to 8 (out of 8 items)

<input type="checkbox"/>	Workspace Name	Default
<input type="checkbox"/>	cite	
<input type="checkbox"/>	it.geosolution	
<input type="checkbox"/>	nurc	
<input type="checkbox"/>	sde	
<input type="checkbox"/>	sf	
<input type="checkbox"/>	tiger	
<input type="checkbox"/>	topp	
<input type="checkbox"/>	venezia	<input checked="" type="checkbox"/>

<< < 1 > >> Results 1 to 8 (out of 8 items)

Creazione di uno store

About & Status

- [Server Status](#)
- [GeoServer Logs](#)
- [Contact Information](#)
- [About GeoServer](#)

Data

- [Layer Preview](#)
- [Workspaces](#)
- [Stores](#)
- [Layers](#)
- [Layer Groups](#)
- [Styles](#)

Services

- [WCS](#)
- [WFS](#)
- [WMS](#)

Settings

- [Global](#)
- [JAI](#)
- [Coverage Access](#)

Tile Caching

- [Tile Layers](#)
- [Caching Defaults](#)
- [Gridsets](#)
- [Disk Quota](#)



Logged in as admin.

Logout

Stores

Manage the stores providing data to GeoServer

[Add new Store](#)

[Remove selected Stores](#)

Results 1 to 10 (out of 10 items)

Search

<input type="checkbox"/>	Data Type	Workspace	Store Name	Type	Enabled?
<input type="checkbox"/>		venezia	agrifood	PostGIS	✓
<input type="checkbox"/>		nurc	img_sample2	WorldImage	✓
<input type="checkbox"/>		nurc	mosaic	ImageMosaic	✓
<input type="checkbox"/>		tiger	nyc	Shapefile	✓
<input type="checkbox"/>		sf	sf	Shapefile	✓
<input type="checkbox"/>		sf	sfdem	GeoTIFF	✓
<input type="checkbox"/>		topp	states_shapefile	Shapefile	✓
<input type="checkbox"/>		topp	taz_shapes	Shapefile	✓
<input type="checkbox"/>		nurc	worldImageSample	WorldImage	✓

Results 1 to 10 (out of 10 items)

Pubblicazione di un layer

About & Status

- Server Status
- GeoServer Logs
- Contact Information
- About GeoServer

Data

- Layer Preview
- Workspaces
- Stores
- Layers
- Layer Groups
- Styles

Services

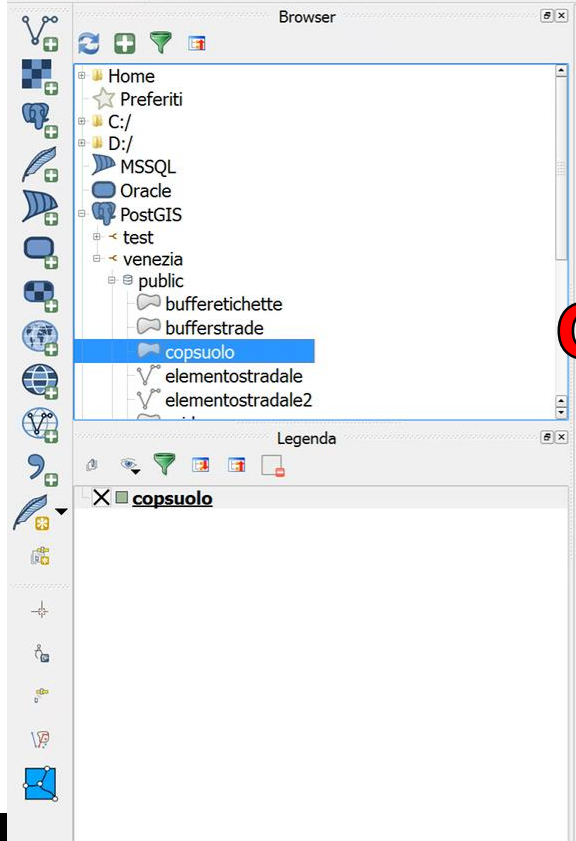
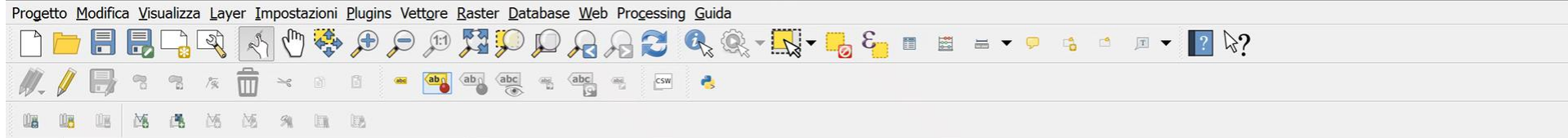
- WCS
- WFS
- WMS

Settings

- Global
- JAI
- Coverage Access

Tile Caching

- Tile Layers
- Caching Defaults
- Gridsets
- Disk Quota



Creazione di uno stile



Sviluppo di un client di visualizzazione

le librerie OpenLayers



A high-performance, feature-packed library for all your mapping needs.

 LATEST

OpenLayers v3.10.1 is here! Check out the [docs](#) and the [examples](#) to get started. The full distribution can be downloaded from the [release page](#).

If you've come here looking for OpenLayers 2.x information, you'll find everything you need on the [2.x page](#).

FEATURES

Tiled Layers

Pull tiles from OSM, Bing, MapBox, Stamen, MapQuest, and any other XYZ source you can find. OGC mapping services and untiled layers also supported.



Fast & Mobile Ready

Mobile support out of the box. Build lightweight custom profiles with just the components you need.



Vector Layers

Render vector data from GeoJSON, TopoJSON, KML, GML, and a growing number of other formats.



Cutting Edge & Easy to Customize

Map rendering leverages WebGL, Canvas 2D, and all the latest greatness from HTML5. Style your map controls with straightforward CSS.



LEARN MORE

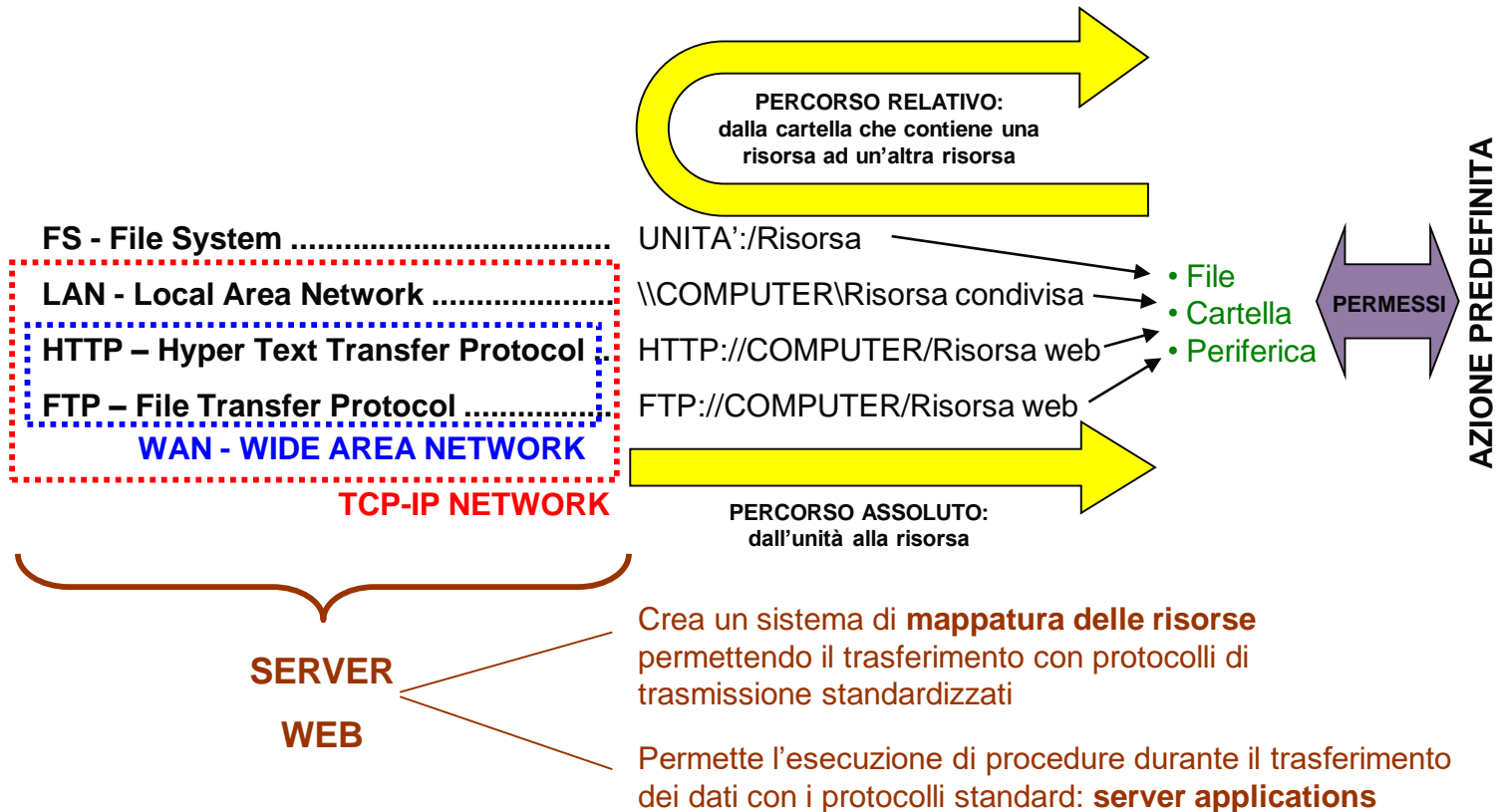
 **Quick Start**

Seen enough already? Go here to get started.

 **Download**

Get the latest release or dig through the archives.

L'accesso alle risorse informatiche e il ruolo dei server web



Programmare “con gli oggetti”

La programmazione si può classificare all'interno di due grandi categorie:

- Programmazione procedurale
- Programmazione ad oggetti

Il codice procedurale viene eseguito sequenzialmente riga per riga ed è composto essenzialmente da assegnazione di valori e istruzioni predefinite.

Il codice ad oggetti è modulare e si basa sull'uso di elementi predefiniti assieme ad elementi definiti dall'utente. Gli elementi (“oggetti” appunto) svolgono sia il ruolo di “contenitori di dati” sia il ruolo di esecutori di determinate azioni.

La programmazione ad oggetti è attualmente la più utilizzata per la sua versatilità e potenza soprattutto in presenza di architetture relativamente complesse.

Programmare “con gli oggetti”

La programmazione ad oggetti si fonda sui concetti di:

- **Classe**
- **Oggetto (ovviamente!)**

Le classi sono i **MODELLI**

Gli oggetti sono **ISTANZE** delle classi

Secondo questo paradigma una classe viene definita una sola volta e con essa è possibile istanziare n oggetti dello stesso tipo.

Mentre nelle CLASSI le caratteristiche sono solo dichiarate,
negli OGGETTI le caratteristiche vengono effettivamente utilizzate

Caratteristiche degli oggetti

Le caratteristiche fondamentali degli oggetti sono

- **Proprietà**
- **Metodi**

E' possibile assimilare un oggetto ad un record di un database:

In questo caso la **tabella è la classe** e viene definita in sql con l'esecuzione della "CREATE TABLE", durante la quale vengono definiti i **campi che sono invece le proprietà** dell'entità rappresentata dalla tabella.

In qualsiasi momento posso inserire un record popolando i campi:

In quest'altro caso l'operazione è simile all'istanziamento di un nuovo oggetto con relativa assegnazione di valore alle proprietà

Caratteristiche degli oggetti

Per quanto riguarda i metodi l'analogia con i DBMS non regge. L'oggetto è infatti una cosa più complessa di un record. Esso non si limita a contenere dati ma **possiede delle capacità di operare** su questi e su altri dati esterni, oltre a poter interagire con altri oggetti.

Le capacità di operare sono delle funzioni denominate **METODI**.

Esempi di sintassi:

Istanziamento:

```
oggetto = new (Classe)
```

Valorizzazione delle proprietà:

```
oggetto.proprietà = valore
```

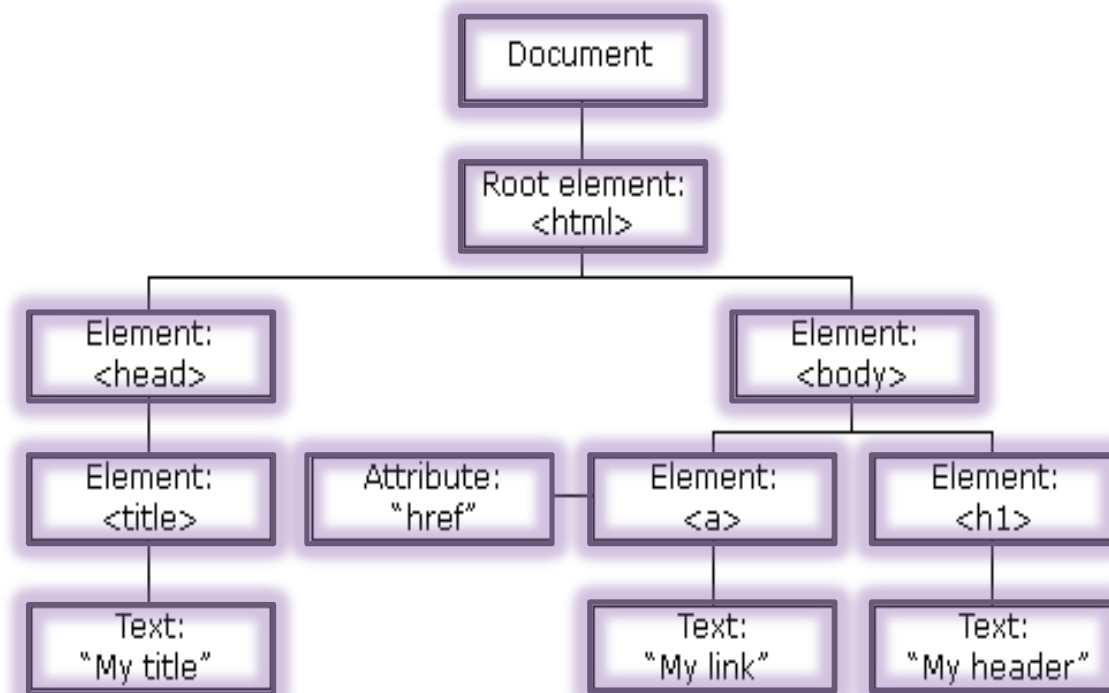
Esecuzione di metodi:

```
oggetto.metodo ()
```

Integrazione tra Javascript e HTML

L'integrazione tra Javascript e pagina HTML avviene mediante l'inserimento di un tag apposito destinato a contenere le righe di codice di programmazione.

La **pagina** viene vista da javascript come un **oggetto** articolato in sotto-oggetti e altri sotto-oggetti ancora secondo il noto schema gerarchico che viene detto tecnicamente **DOM** (Document Object Model):

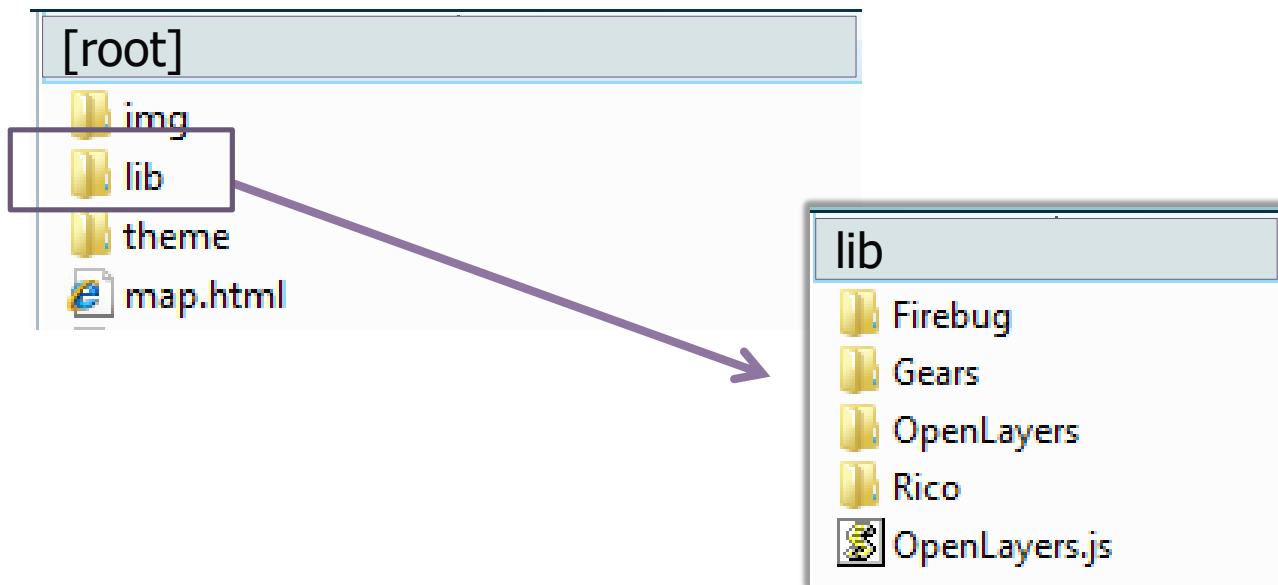


Pagina HTML standard

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Esempio OpenLayers</title>
  </head>
  <body>
    <h3 id="title">Mappa con OpenLayers</h3>
    <div id="docs">
      Semplice mappa con un layer WMS e controlli di base.
    </div>
  </body>
</html>
```

Collegamento alle librerie e agli stili

```
<link rel="stylesheet" href="theme/default/style.css" type="text/css" />  
<script src="lib/OpenLayers.js"></script>
```



Quick Start - put a map on a page

```
<!doctype html>
<html lang="en">
  <head>
    <link rel="stylesheet" href="http://openlayers.org/en/v3.10.1/css/ol.css" type="text/css">
    <style>
      .map {
        height: 400px;
        width: 100%;
      }
    </style>
    <script src="http://openlayers.org/en/v3.10.1/build/ol.js" type="text/javascript"></script>
    <title>OpenLayers 3 example</title>
  </head>
  <body>
    <h2>My Map</h2>
    <div id="map" class="map"></div>
    <script type="text/javascript">
      var map = new ol.Map({
        target: 'map',
        layers: [
          new ol.layer.Tile({
            source: new ol.source.MapQuest({layer: 'sat'})
          })
        ],
        view: new ol.View({
          center: ol.proj.fromLonLat([37.41, 8.82]),
          zoom: 4
        })
      });
    </script>
  </body>
</html>
```

Struttura del viewer

To include a map a web page you will need 3 things:

1. Include OpenLayers
2. `<div>` map container
3. JavaScript to create a simple map

Include OpenLayers

```
<script src="http://openlayers.org/en/v3.10.1/build/ol.js" type="text/javascript"></script>
```

The first part is to include the JavaScript library. For the purpose of this tutorial, here we simply point to the openlayers.org website to get the whole library. In a production environment, we would build a custom version of the library including only the module needed for our application.

`<div>` to contain the map

```
<div id="map" class="map"></div>
```

The map in the application is contained in a `<div>` HTML element. Through this `<div>` the map properties like width, height and border can be controlled through CSS. Here's the CSS element used to make the map 400 pixels high and as wide as the browser window.

```
<style>
  .map {
    height: 400px;
    width: 100%;
  }
</style>
```


Struttura del viewer

```
var map = new ol.Map({
  target: 'map',
  layers: [
    new ol.layer.Tile({
      source: new ol.source.MapQuest({layer: 'sat'})
    })
  ],
  view: new ol.View({
    center: ol.proj.fromLonLat([37.41, 8.82]),
    zoom: 4
  })
});
```

With this JavaScript code, a map object is created with a MapQuest Open Aerial layer zoomed on the African East coast. Let's break this down:

The following line creates an OpenLayers `Map` object. Just by itself, this does nothing since there's no layers or interaction attached to it.

```
var map = new ol.Map({ ... });
```

To attach the map object to the `<div>`, the map object takes a `target` into arguments. The value is the `id` of the `<div>`:

```
target: 'map'
```

The `layers: [...]` array is used to define the list of layers available in the map. The first and only layer right now is a tiled layer:

```
layers: [
  new ol.layer.Tile({
    source: new ol.source.MapQuest({layer: 'sat'})
  })
]
```

Struttura del viewer

Layers in OpenLayers 3 are defined with a type (Image, Tile or Vector) which contains a source. The source is the protocol used to get the map tiles. You can consult the list of [available layer sources here](#)

The next part of the `Map` object is the `View`. The view allow to specify the center, resolution, and rotation of the map. The simplest way to define a view is to define a center point and a zoom level. Note that zoom level 0 is zoomed out.

```
view: new ol.View({
  center: ol.proj.fromLonLat([37.41, 8.82]),
  zoom: 4
})
```

You will notice that the `center` specified is in lon/lat coordinates (EPSG:4326). Since the only layer we use is in Spherical Mercator projection (EPSG:3857), we can reproject them on the fly to be able to zoom the map on the right coordinates.



venezia.htm

```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <link rel="stylesheet" href="http://openlayers.org/en/v3.1.1/css/ol.css" type="text/css">
5   <style>
6     .map {
7       height: 500px;
8       width: 600px;
9     }
10  </style>
11  <script src="http://openlayers.org/en/v3.1.1/build/ol.js" type="text/javascript"></script>
12  <title>OpenLayers 3 example</title>
13 </head>
14 <body>
15   <div id="map" class="map"></div>
16   <script type="text/javascript">
17     var map = new ol.Map({
18       target: 'map',
19       layers: [
20         new ol.layer.Tile({
21           source: new ol.source.MapQuest({ layer: 'sat' })
22         }),
23         new ol.layer.Tile({
24           source: new ol.source.TileWMS(/** @type {olx.source.TileWMSOptions} */({
25             url: '.....',
26             params: { 'LAYERS': '.....', 'TILED': true },
27             serverType: 'geoserver'
28           })))
29       ],
30     },
31     view: new ol.View({
32       center: ol.proj.transform([12.5, 45.5], 'EPSG:4326', 'EPSG:3857'),
33       zoom: 10
34     })
35   </script>
36 </body>
37 </html>
```

Creazione del web viewer