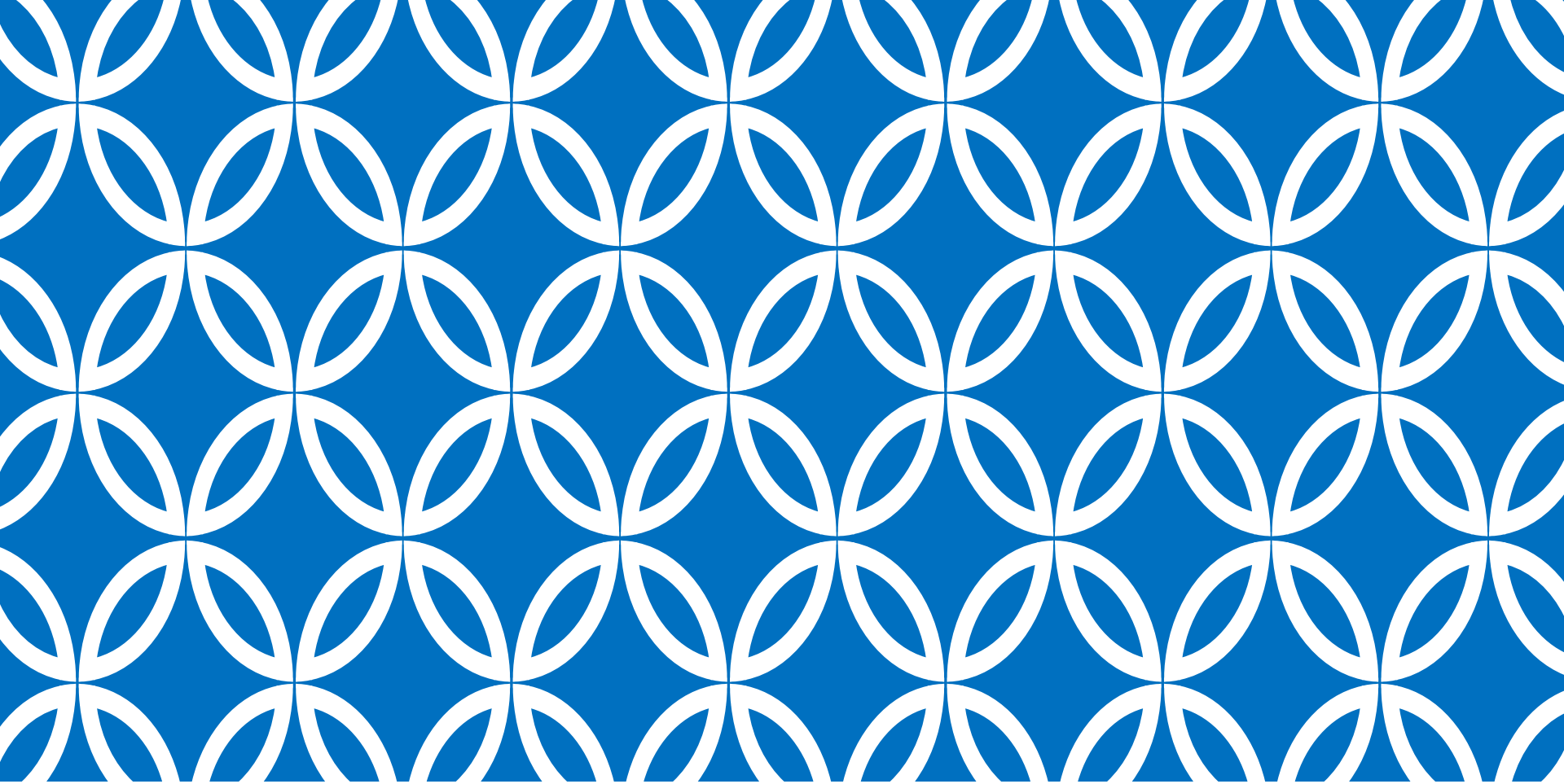
LEARN. DO. EARN

ACADGILD

FRONT END DEVELOPMENT (WITH ANGULARJS)

Website : http://www.acadgild.com
LinkedIn : https://www.linkedin.com/company/acadgild
Facebook : https://www.facebook.com/acadgild



Session 17 – AngularJS $http

Agenda – AngularJS $http

1. Server Communication - $http
2. Configuring Request
3. Understanding Parameters
4. $http Function Shortcuts
5. Caching Server Response
6. Custom Transformation
7. Request & Response Transformation
8. Handling Restful Resources
9. Using ngResource API
10. Using The Returned Restful Object

Agenda – AngularJS $http

11. Caching In AngularJS
12. Using Cache Object
13. $http Caching
14. Using $http Cache Within Code
15. Using Custom Cache For $http
16. Configuring Cache Settings Using $http

Server Communication - $http

Core Angular service that facilitates communication with the remote HTTP servers via the browser's XMLHttpRequest object or via JSONP.

```
$http({method: 'GET', url: '/someUrl'})
  .success(function(data, status, headers, config) {
    // this callback will be called asynchronously
    // when the response is available
  })
  .error(function(data, status, headers, config) {
    // called asynchronously if an error occurs
    // or server returns response with an error status.
  });
```

Success asynchr

Configuring Request

- Add some authorization headers
- Setting cache
- Transforming the request or response

```
$http({
  method: string,
  url: string,
  params: object,
  data: string or object,
  headers: object,
  transformRequest: function transform(data, headersGetter),
  transformResponse: function transform(data, headersGetter),
  cache: boolean or Cache object,
  timeout: number,
  withCredentials: boolean
});
```

Understanding Parameters

- **method:** The method represents the method of communicating to the server. It can be GET,POST,DELETE etc
- **url:** The URL parameter represents the URL of the server.
- **params:** The parameters to send with the request.
- **data:** The data to be sent with the request. Ex post data.
- **Headers:** Any additional headers to be sent with the request. **WithCredentials:** Setting this parameter to true will also send the session id with the request.
- **transformRequest:** The function to modify the request
- **transformResponse:** The function to modify the server response
- **cache:** Boolean parameter for caching or custom caching object reference
- **timeout:** The number to represent the number of milliseconds after which request will timeout

$http Function Shortcuts

- Angular provides a list of functions which are shortcuts for making different types of requests.
  - **$http.get :** for making get request
  - **$http.head:** for making head request
  - **$http.post :** post requests shortcut
  - **$http.put:** put request shortcut
  - **$http.delete :** delete request shortcut
  - **$http.jsonp :** for JSONP requests
  - **$http.patch :** For making patch requests

- **Following is the syntax to use the functions:**

```
$http.get('/serverUrl').success(successCallback);
$http.post('/serverUrl', data).success(successCallback);
```

Caching Server Response

To implement caching we can set the caching parameter value to be true.

Syntax:

```
$http.get('http://server/api', {
cache: true
}).success(function() { //success handling });
```

Custom Transformation

- **If we want to add our own transformation**, then we can pass in functions as part of the config.

- We can transform a request or a response at the request/response phase using the **transforRequest** and **transformResponse property** of the object that we pass to the **$http function**.

- **Note:** that this method is good if we want to modify a single request/response.

- **If we want to modify the request/response globally**, we need to do it in the config function of the module.

Request & Response Transformation

- Request and response transformation is the modification of request or response that is made by angular.

- Angular provides few inbuilt request and response transformations

- **Request transformations:**
- If the data property of the config request object contains an object, angular serializes it into JSON format.

- **Response transformations:**
- If an XSRF prefix is detected in the response, angular strips it.
- If a JSON response is detected, angular deserializes it using a JSON parser

Handling Restful Resources

- $http provides a great API for communicating with server.
- To have the restful API, we need to use the ngResource Module.
- **The ngResource** module provides a well built API to handle rest based calls.
- To use it we have to download the angular-resource.js file.
- While declaring the app we have to include ngResource as dependency.

  Example:

  Var app = angular.module('myApp',[ngResource]);

Using ngResource API

- To create a resource object we need use the $resource service.

  var resourceOb = $resource('api/comments/:id');

  Here the **:** id represents the dynamic parameter id for the REST request.
- **'get':** {method:'GET'}, 'save': {method:'POST'},
- **'query':** {method:'GET', isArray:true}, 'remove': {method:'DELETE'}, 'delete': {method:'DELETE'}
- **To call a get method we just need to write:**

  Var resultOb= Entry.get({id:12},function(){

  – //have success handler here

  });

Using The Returned Restful Object

- When data is returned from the server, the returned object is also an instance of the resource class.

- The actions save, remove and delete are available on this object as methods with $ prefix.

- This technique allows us to easily perform CRUD operations on the returned object itself.

Caching In AngularJS

- The **$cacheFactory** service generates cache objects for all Angular services. We can create a cache object by

  ```
  var appCache = $cacheFactory('AppCacheID');
  ```

- Here **AppCacheID** is the id of appCache.

- **Note** that we can pass the optional object to the $cacheFactory where we can specify the number of cache objects it can store.

Using Cache Object

	Description	Example
info()	This method returns ID, size and options of the cache object.	**appCache.put('name', 'sumit');**
put()	This method stores the data in the cache, The first parameter is the key and second is the data to store	**appCache.put('hobby', 'coding');**
get()	get methods retrieves the data from the cache for the given cache id	**appCache.get('name');**
remove()	This function removes a key-value pair from the cache, if it's found. If not found, it returns undefined	**appCache.remove('name');**
removeAll()	function resets the cache and removes all cached values	**appcache.removeAll()**
destroy()	This method removes all references of the current cache from the $cacheFactory cache registry.	**appCache.destroy()**

$http Caching

- The $http service creates a cache with the ID $http. To enable the $http request to use this default cache object we have to pass a cache parameter for $http service.
- **Example:**

```
$http({ method: 'GET',
url: '/api/comments.json', cache: true
});
```

- **Note** for every request the key for the $http cache is the full-path URL.

Using $http Cache Within Code

- Usually we don't need to use the $http cache object but if need we can retrieve it using

  var cache = $cacheFactory.get('$http');
- **//To retrieve the cache object for a particular url**

  var commentsCache =cache.get('http://acadgild.com/api/comments.json');
- **//To Delete the cache entry for the previous request**

  cache.remove('http://acadgild.com/api/comments.json');
- **//To remove the entire cache**

  cache.removeAll();

Using Custom Cache For $http

Instead of passing a boolean true for $http, we can pass the instance of the cache.

Example:

```
var appCache = $cacheFactory.get('AppCacheID');
$http({ method: 'GET‘,
url: '/api/comments.json', cache: appCache
});
```

Configuring Cache Settings Using $http

- Configure the setting for cache for $http .

- Below is the snippet:

```
angular.module('CommentApp‘)
.config(function($httpProvider) {
 $httpProvider.defaults.cache = $cacheFactory('appCache',
 {capacity: 30}); } );
```

Lets Discuss Assignments