Creating the Article-Fetching System in Typescript

- Adding the CSS needed for the site
- Creating a Document Helper to assist in DOM element creation
- Creating a frontend Article Manager
- Adding an interface for the Article Info

main.css

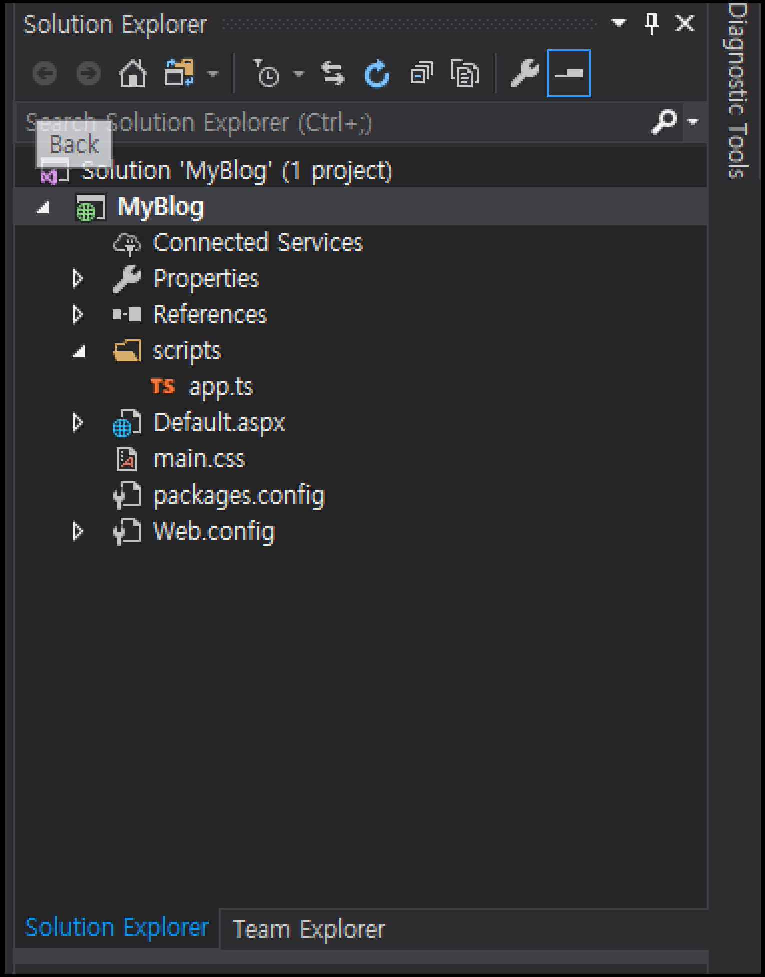
```css
htmnl,body {
   font-family: Arial, sans-serif;
   background-color:black;
   color: #444;
   font-size: 14pt;
}
a, a:visited {
   text-decoration: none;
   color: #ff6600;
}
.banner {
   font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;
   font-size: 40pt;
   width: 100%;
   clear: both;
   text-align:center;
   font-weight:bold;
   border-bottom:1px solid #888;
}
```

```
.article {
   border-bottom: 1px solid #888;
   margin: 20px auto;
   min-width:400px;
   clear: both;
}

.article-title {
   font-weight:bold;
   font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida
Grande', 'Lucida Sans', Arial, sans-serif;
   color:white;
}

.article-content {
   font-family: Georgia, 'Times New Roman', Times, serif;

}
```



```
Solution 'MyBlog' (1 project)
  MyBlog
    Connected Services
    Properties
    References
    scripts
      app.ts
    Default.aspx
    main.css
    packages.config
    Web.config
```

documentHelper.ts

```typescript
module Blog {
    /**
     * A utility class to assist with the creation of document elements.
     **/
    export class DocumentHelper {

        /**
         * Creates an article entry using the provided info. Returns an element.
         * @param articleInfo The article info to create the entry from.
         **/
        public static createArticleEntry(articleInfo: IArticleInfo): HTMLDivElement {
            let wrapper = DocumentHelper.createDiv("article");
            wrapper.id = "article_" + articleInfo.id;

            let header = DocumentHelper.createHeader(2, articleInfo.Title, "article-title");
            wrapper.appendChild(header);

            let content = DocumentHelper.createDiv("article-content");
            content.innerHTML = articleInfo.Content;
            wrapper.appendChild(content);

            return wrapper;
        }
/**
         * Creates a div element.
         * @param className Optional class name to be applied to created div
         **/
        private static createDiv(className?: string): HTMLDivElement {
            let element = document.createElement("div") as HTMLDivElement;
            if (className) {
                element.className = className;
            }
            return element;
        }
        /**
         * Creates a header element with the specified level (i.e. h1,h2,)
         * @param level The level of element to create [1-6]
         * @param text The text to add to the element.
         * @param className Optional class name to be applied to create element.
         **/
        private static createHeader(level: number, text: string, className?: string) {
            let element = document.createElement("h" + level) as HTMLHeadingElement
            element.innerHTML = text;
            if (className) {
                element.className = className;
            }
            return element;
        }
    }
}
```

IArticleInfo.ts

```
module Blog {
    /**
     * Represents an article in the system.
     */
    export interface IArticleInfo {
        /** The article identifier. */
        Id: number;

        /** The article title. */
        Title: string;

        /** The content of the article. */
        Content: string;
    }
}
```

ArticleManager.ts

```
module Blog {
    /**
     * Manages the retrieval of articles in the system.
     */
    export class ArticleManager {
        private _articleContainer: HTMLDivElement;
        private _lastLoadedArticleId: number;

        /**
         * Creates a new article manager.
         * @param elementId The identifier of the HTML element to insert into
         */
        public constructor(elementId: string) {
            this._articleContainer = document.getElementById(elementId) as HTMLDivElement;
            if (this._articleContainer === undefined) {
                throw new Error("Unable to find element named: " + elementId)
            }
        }
```

```
/**
 * Called when an article request has loaded.
 * @param request The request object.
 **/
private onArticleLoaded(request: XMLHttpRequest) {
    if (request.readyState === XMLHttpRequest.DONE) {
        let responseText = request.responseText.trim();
        if (responseText === "") {
            throw new Error("Error parsing response.");
        }
        // Parse the JSON response.
        let articleInfo: IArticleInfo = JSON.parse(responseText) as IArticleInfo;
        //Set the last post loaded marker.
        this._lastLoadedArticleId = articleInfo.Id;
        //Create the post elements and add it to the page.
        let articleElement = DocumentHelper.createArticleEntry(articleInfo);
        if (articleElement !== undefined) {
            this._articleContainer.appendChild(articleElement);
        }

    }
}
```

```
/** * Called when there is an error loading an article.
 * @param request The request object. **/
private onArticleLoadError(request: XMLHttpRequest): void {
    console.warn("Error loading post: " + request.status + " - " + request.statusText);
}
/**Called when the article request is aborted.
 * @param request The request object. **/
private onArticleLoadAborted(request: XMLHttpRequest): void {
    console.warn("Article load aborted.");
}
/*** Loads the next article in the system.*/
public loadNextArticle(): void {
    let request = new XMLHttpRequest();
    request.addEventListener("readystatechange", this.onArticleLoaded.bind(this,request));
    request.addEventListener("error", this.onArticleLoadError.bind(this,request));
    request.addEventListener("abort", this.onArticleLoadAborted.bind(this,request));
    let url = "GetNextArticle.aspx";
    if (this._lastLoadedArticleId !== undefined) {
         url += "?lastArticleId=" + this._lastLoadedArticleId;
        }
request.open("GET", url, true);
request.send(null);
}
}
}
```

app.ts

```
//The article manager.
var articleManager: Blog.ArticleManager;

//Entry point
window.addEventListener("load", function () {
    articleManager = new Blog.ArticleManager("articles");
    articleManager.loadNextArticle();
});
```