



IBM Carbon TreeView

Example 02



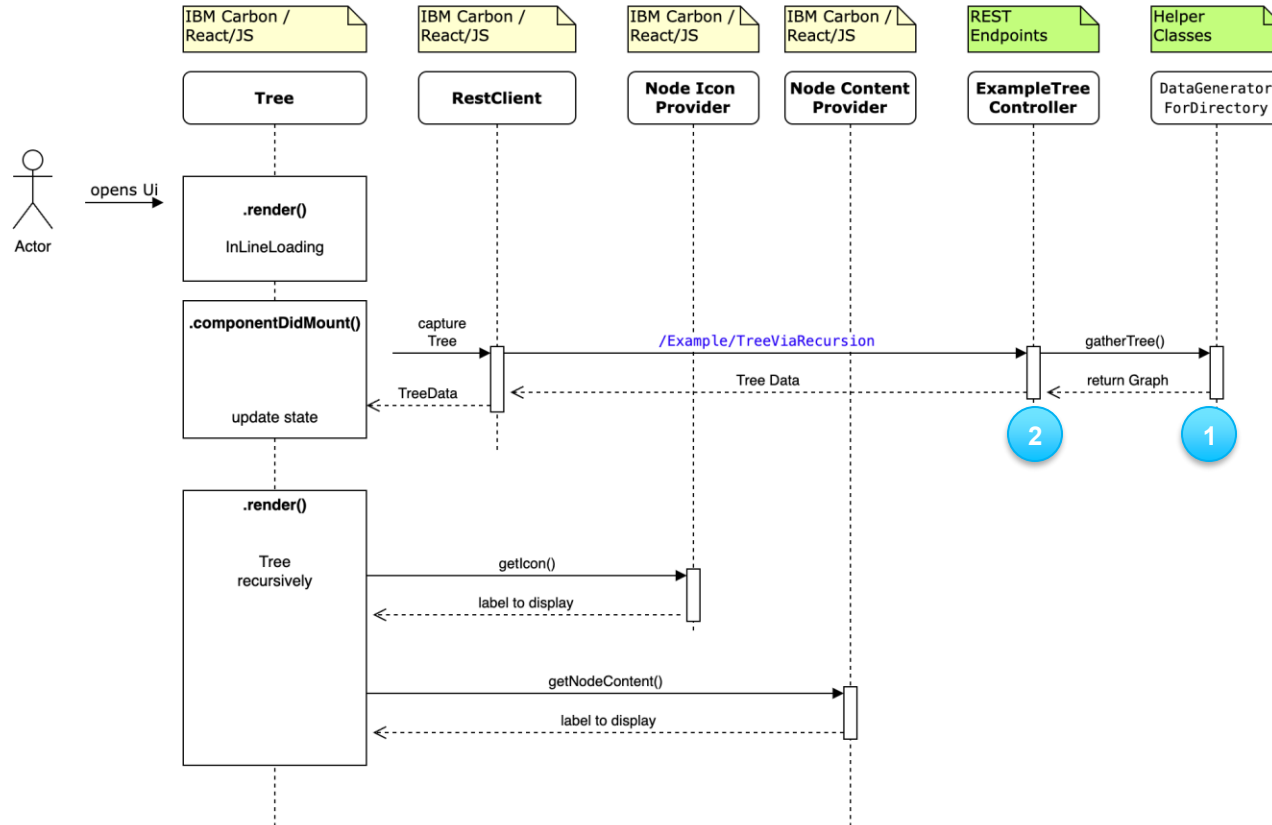
Tree (filled from Explain-Directory)

GET

/api/carbonplay/Explain/Package/Deliver/{identification}







Sequence Diagram





Server-sided Sources

1

- ▼  restserver.objects.tree.generator
 - >  DataGeneratorForDirectory.java
 - >  SimpleDummyDataGenerator.java
 - >  SimpleDummyDataGeneratorViaList.java

2

- ▼  restserver.controller
 - >  AppConfigController.java
 - >  App18nController.java
 - >  AppUiParameterController.java
 - >  ExampleTreeController.java



1

Start with RootPath

```
/**
 * This method creates a tree based on a Directory as startPoint and delivers the content as Tree
 * @param p Starting Point of Directory Structure
 * @return
 * @throws Exception
 */
public static SimpleDirectedGraph<PojoJGraphtTreeNode, DefaultEdge> gatherEntitiesTree(Path p) throws Exception {

    // =====
    // a) create tree as graph (Vertices & Edges) recursively
    // =====

    SimpleDirectedGraph<PojoJGraphtTreeNode, DefaultEdge> graph = new SimpleDirectedGraph<>(DefaultEdge.class);

    PojoJGraphtTreeNode pojoRoot = new PojoJGraphtTreeNode();
    walkDirectory(graph, p, p.toFile(), pojoRoot);

    // =====
    // b) calculate parent and children
    // =====

    JGraphTParentChildrenPopulator.calcParentIDAndChildrens(graph);

    return graph;
}
```

Recursion

```
/**
 * This method walks through files in root-directory recursively
 * @param graph SimpleDirectedGraph to populate
 * @param root Root-Directory (with prefix which is always to be removed from filename)
 * @param f current directory or file to handle
 * @param pojo PojoTreeNode to populate
 */
private static void walkDirectory(SimpleDirectedGraph<PojoJGraphtTreeNode, DefaultEdge> graph, Path root, File f, PojoJGraphtTreeNode pojo)

    populatePojoWithFileData(root, f, pojo);
    graph.addVertex(pojo);

    File[] files = f.listFiles();
    if (files == null) return; // Could be due to permission issues

    //walk through childs
    for (File fChild : files) {

        PojoJGraphtTreeNode pojoChild = new PojoJGraphtTreeNode();

        populatePojoWithFileData(root, fChild, pojoChild);
        graph.addVertex(pojoChild);

        graph.addEdge(pojoChild, pojo);

        if (fChild.isDirectory()) {
            walkDirectory(graph, root, fChild, pojoChild); // Recurse into sub-directory
        }
    }
}
```



1

Populate
Node

```
/**
 * This method populates a PojoJGraphtTreeNode with
 * <ul>
 * <li>Payload File</li>
 * <li>Payload UiControl</li>
 * </ul>
 * @param root Root-Directory (with prefix which is always to be removed from filename)
 * @param f File relative to RootPath
 * @param pojo to populate
 */
private static void populatePojoWithFileData(Path root, File f, PojoJGraphtTreeNode pojo) {

    String id = f.getAbsolutePath();
    id = id.replaceFirst(root.toString(), "");
    id = id.replaceFirst(File.separator, "");

    //id = only relative path to root
    String input = f.getAbsolutePath();
    String marker = root.toString();
    int index = input.indexOf(marker);
    String label = (index != -1) ? input.substring(index + marker.length()) : input;
    String[] explode = StringUtils.explode(id, "/");
    if (explode.length != 0) {
        label = explode[explode.length-1];
    }

    pojo.setId(id);
    pojo.setLabel(label);
    pojo.setValue("");
    pojo.setChildren(new PojoJGraphtTreeNode[0]);
    PojoFilePayload pojoFilePayload = createPojoFilePayload(f);
    pojo.setPayload(pojoFilePayload);

    PojoPayloadUiControl pojoPayloadUiControl = new PojoPayloadUiControl();
    pojo.setUiControl(pojoPayloadUiControl);
}
```

Create
Payload

```
/**
 * FrontEnds like Carbon require real Pojos to access File's information
 * This method transfers required File Attributes for a file to a Pojo
 * @param pFile Path to File
 * @return PojoPayloadFile
 */
private static PojoFilePayload createPojoFilePayload(File f) {
    PojoFilePayload pojoFilePayload = new PojoFilePayload();
    pojoFilePayload.setFlagExists(f.exists());
    if (f.exists()) {
        pojoFilePayload.setFlagCanRead(f.canRead());
        pojoFilePayload.setFlagCanWrite(f.canWrite());
        pojoFilePayload.setFlagIsDirectory(f.isDirectory());
        pojoFilePayload.setFlagIsFile(f.isFile());
        pojoFilePayload.setFlagIsHidden(f.isHidden());
        pojoFilePayload.setLastModified(f.lastModified());
        pojoFilePayload.setLength(f.length());
    }
    return pojoFilePayload;
}
```



```
//===== >

//don't worry to much about this ... it is just a way to gain access to explain_packages
//in your concrete situation you can choose any java.nio.file.Path as pRoot
ServiceLocator sl = ServiceFactory.getInstance().getServiceLocator();
IAppDirectoryService appDirectoryService = sl.getAppDirectoryService();
Path pRoot = appDirectoryService.getAppDirectory("explain_packages");

//get data as Graph without any edges
SimpleDirectedGraph<PojoJGraphTreeNode, DefaultEdge> graph = DataGeneratorForDirectory.gatherTree(pRoot);

//identify root-nodes
PojoJGraphTreeNode[] array = JGraphTListToTreeTransformer.getTreeRootNodes(graph);

//<=====

result = new ResponseEntity<>(array,HttpStatus.OK);
return result;
```



Client-sided Sources

▼ Tree_02_BackendFilledExplainDirectory

▼ backend

RESTClient.ts

▼ uihelper

NodeContentProvider.js

NodeIconProvider.js

Tree.js

PojoJGraphtTreeNode.ts



```
export type PojoPayloadUiControl = {
  visible : boolean,
  expanded : boolean,
}

export type PojoJGraphtTreeNode = {
  id: string;
  label: string;
  value: string;
  parentId?: string //id of parent (choose null for 'no Parent')
  payload?: any;    //Payload (typically a Pojo ins JSON-Format), can be null
  uicontrol?: PojoPayloadUiControl; //Payload to control ui
  children: PojoJGraphtTreeNode[];
}

export type PojoJGraphtTreeNodeList = PojoJGraphtTreeNode[]
```