HealthCheck-v3.0

1 系统界面

如图 1.1-1.4 所示为系统界面，包括 Home 页面，Features 页面（Health Check 和 Performance Compare）和 Contact 页面。

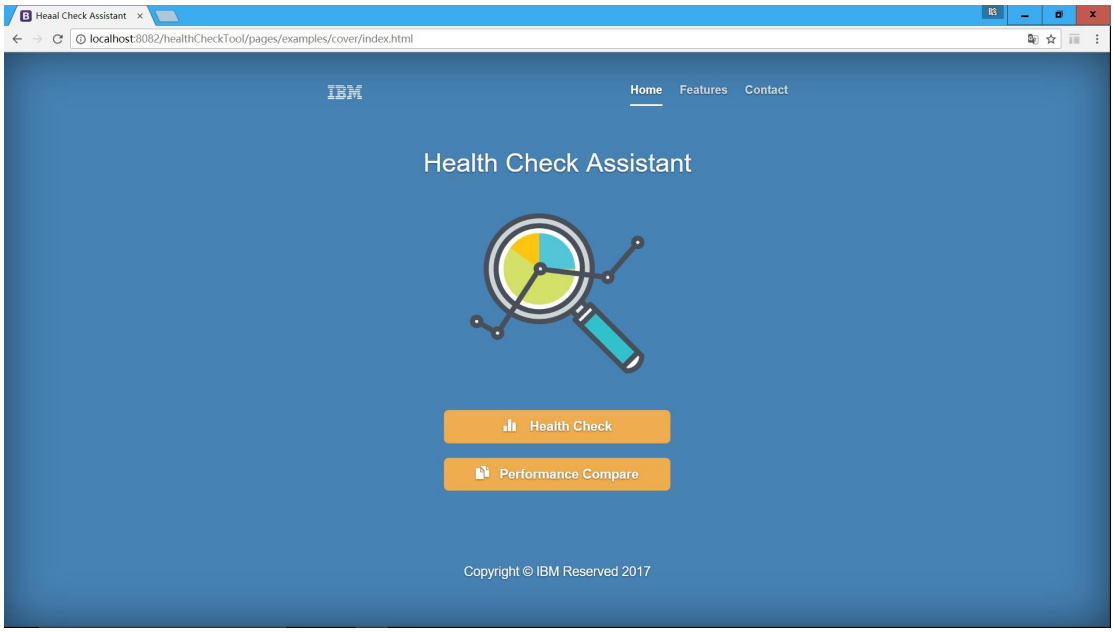


图 1.1 首页

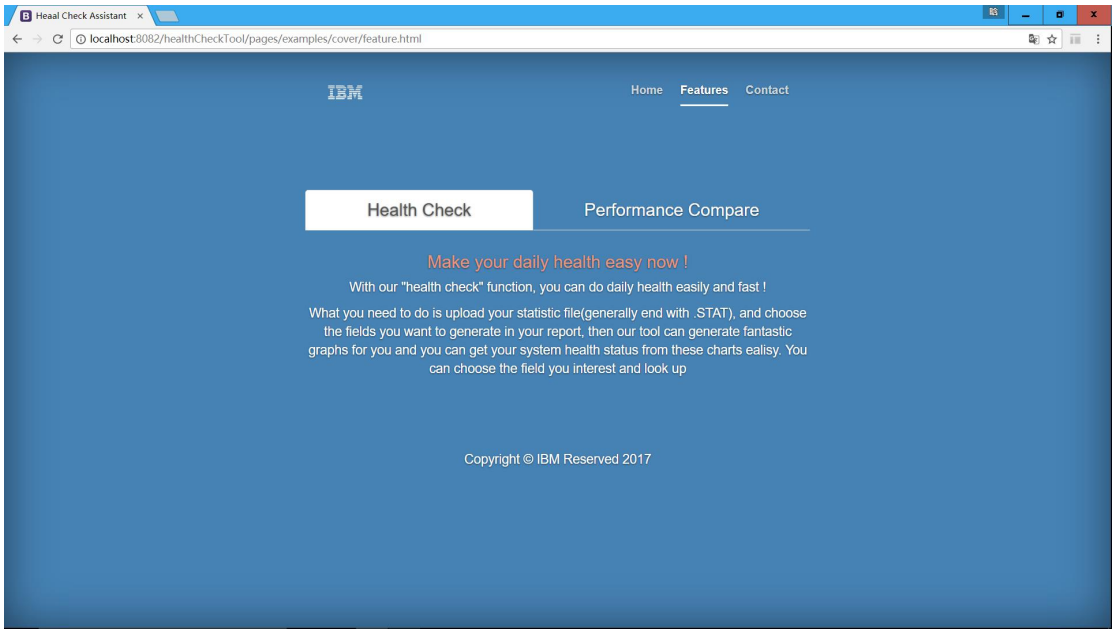


图 1.2 HealthCheck Features

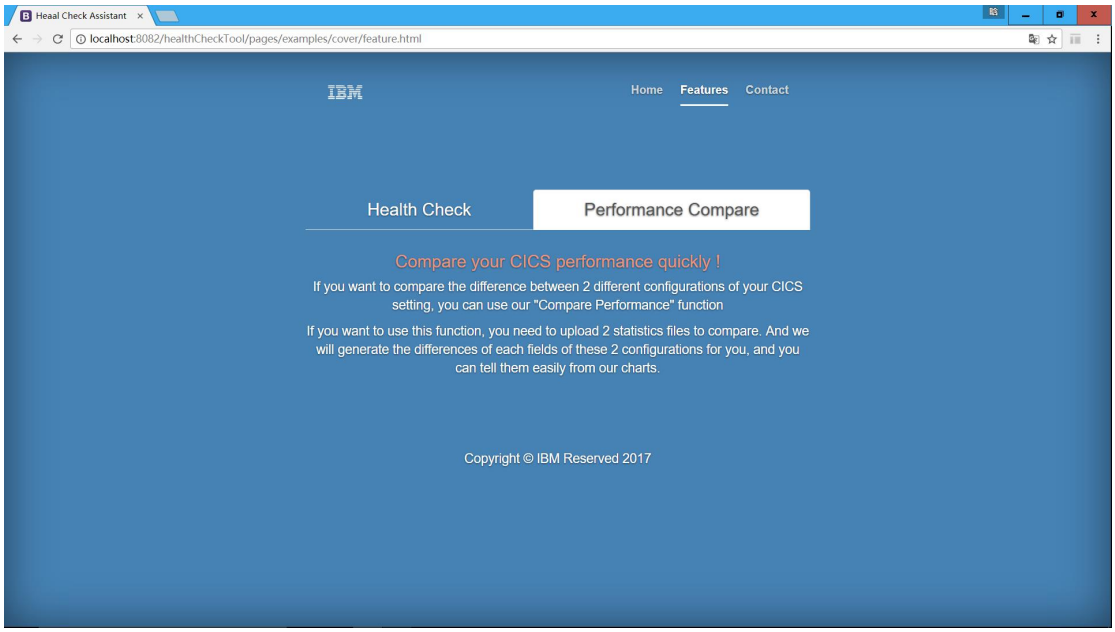


图 1.3　Performance Compare Features

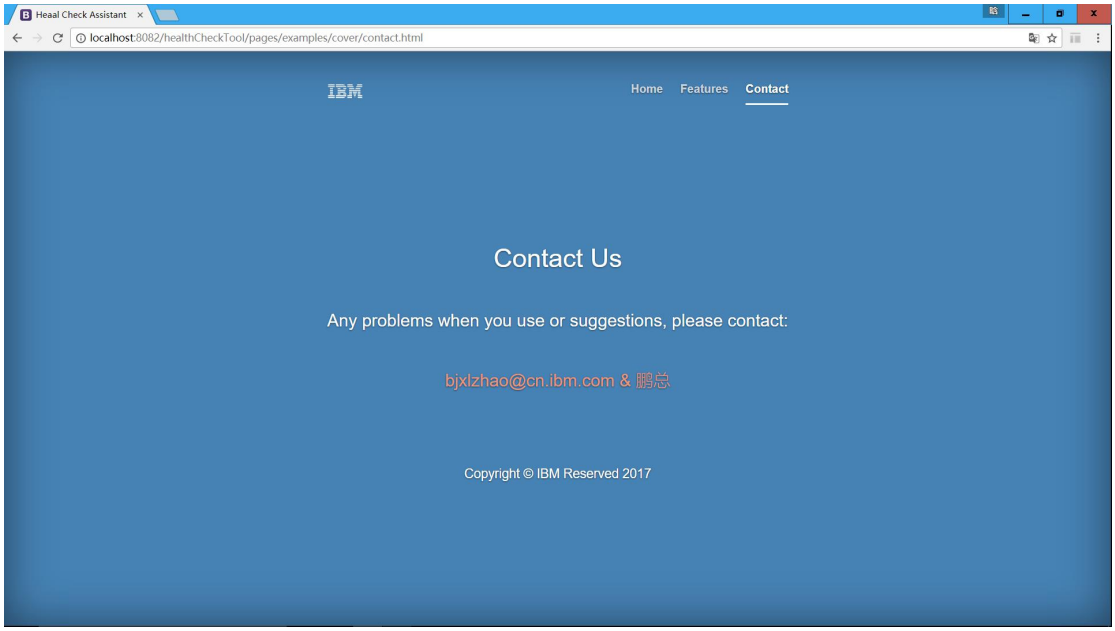
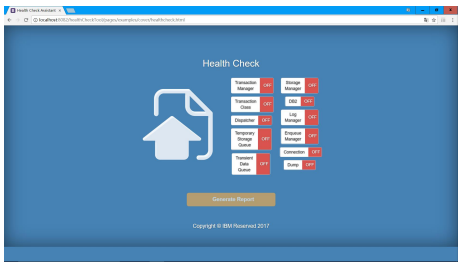
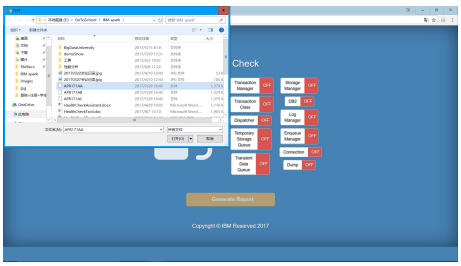


图 1.4　Contact

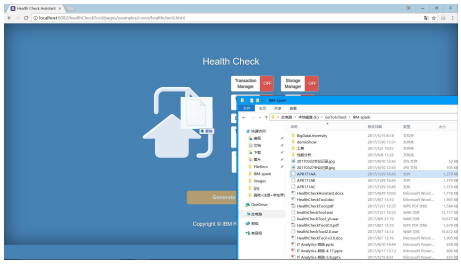
2　健康检查



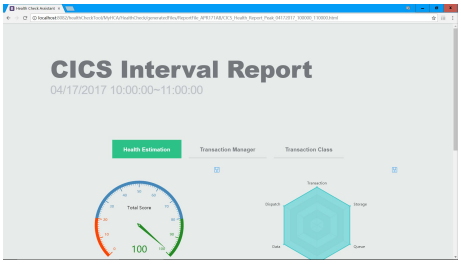
a 初始状态



b 选择上传



c 拖拽上传



d 显示结果

图 2.1　健康检查

如图 2.1 所示，采用选择上传和拖拽上传两种方式上传日志文件，后台对上传的日志文件用 python 脚本处理，生成健康检查报告。

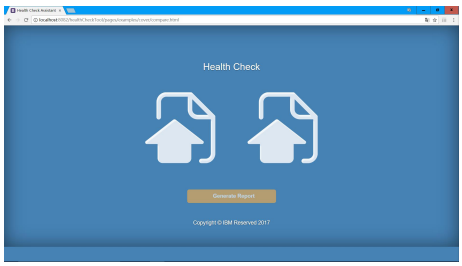
a 为初始状态，button 为不可点击状态。

b 为选择上传，通过 onchange 事件获取上传文件

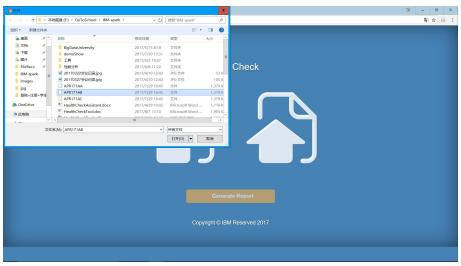
c 为拖拽上传，通过 drop 事件获取上传文件

d 为点击生成按钮，将文件通过 ajax 以 formData 的方式上传给后端。后端调用 python 脚本进行健康检查，生成的结果。

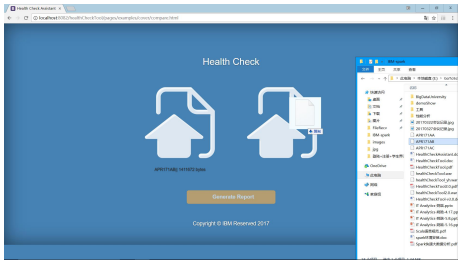
3　配置对比



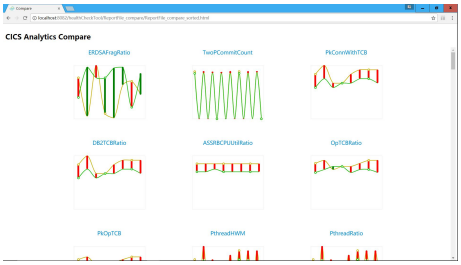
a 初始状态



b 点击上传



c 拖拽上传



d 显示结果

图 3.1　配置对比

如图 3.1 所示，为配置对比页面。通过点击上传和拖拽上传两种方式将配置前和配置后的日志文件进行上传。后台调用 python 脚本，先对日志文件进行 HealthCheck 操作，生成 rawData 数据，再对两个 rawData 数据进行性能对比，生成性能对比报告。进行对比操作，生成比较结果。

a 为初始状态，button 为不可点击状态。

b 为选择上传，通过 onchange 事件获取上传文件

c 为拖拽上传，通过 drop 事件获取上传文件

d 为点击生成按钮，将文件通过 ajax 以 formData 的方式上传给后端。后端调用 python 脚本进行健康检查，生成的结果。

4 项目结构

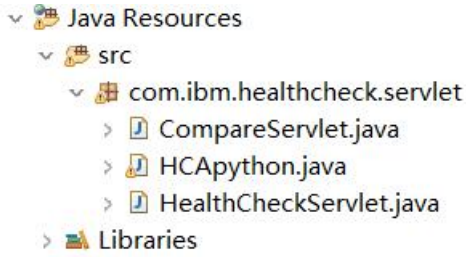


图 4.1 java 代码

如图 4.1 所示为 java 代码。HealthCheckServlet.java 是 Health Check 部分的上传文件部分。上传成功后跳转到 HCApython 调用 MyHCA.py 生成健康检查报告。CompareServlet 是 Performance 部分，获取上传的两个日志文件，通过 MyHCA.py 生成健康检查报告，将两个报告的 RawData 文件夹进行对比生成性能对比报告。

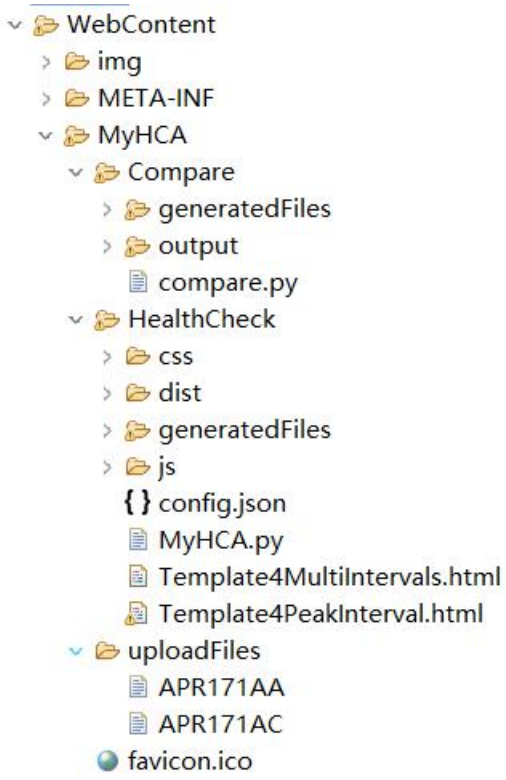


图 4.2 MyHCA 文件

如图 4.2 为具体的业务逻辑文件,在 WebContent/MyHCA 路径下。包括 Compare 文件夹，HealthCheck 文件夹和 uploadFiles 文件夹。

所有上传的文件都在 uploadFiles 路径下，

HealthCheck 的 MyHCA.py 文件生成的健康检查报告在 HealthCheck/generatedFiles 路径下，命名规则为 ReportFile_文件名。

Compare 的 compare.py 生成的性能对比报告在 Compare/generatedFiles 下。命名规则为 ReportFile_compare_文件名 1_文件名 2。

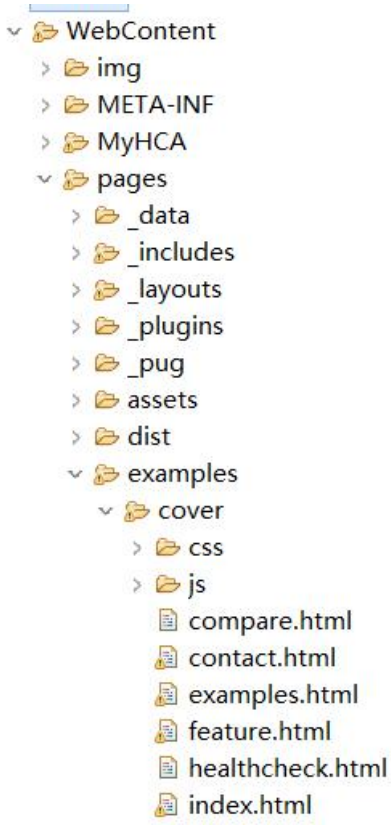


图 4.3 HTML 文件

如图 4.3 所示为 HTML 文件，在 WebContent/pages/examples/cover 路径下。css 和 js 分别为页面的样式表和脚本文件。

index.html 是首页。

feature.html 是特征页。

Contact.html 是联系页。

Healthcheck.html 是健康检查页。

compare.html 是性能对比页。

5 具体实现

5.1 文件上传

文件上传分为点击上传和拖拽上传两种方式。

1. 点击上传采用 html 的 input 标签，监听 input 的 onchange 事件获得上传的文件信息。

```
<div class="image carousel-inner">
    <input type="file" name="file" id="file" class="image-input"
    ondragenter="    return false" onchange="selectFile()" />
    <img src="../../upload_file.png" style="width: 280px" id="first-file" />
</div>
function selectFile() {
    var f = document.getElementById("file").files;
    uploadFile = f[0];
    show(f[0])
    $("#check").attr("disabled", false)
}
```

2. 拖拽上传采用 html5 的 drag&drop API。获得上传的文件信息。

```
var body = document.getElementById('body');
body.addEventListener('dragover', handleDragOver, false);
body.addEventListener('drop', handleFileSelect, false);
/*处理拖拽文件*/
function handleFileSelect(evt) {
    /*屏蔽默认拖拽打开文件*/
    evt.stopPropagation();
    evt.preventDefault();
    var files = [], items = evt.dataTransfer.items;/*拖拽的文件*/
    for (var i = 0; i < items.length; i++) {
        var entry = items[i].webkitGetAsEntry();
        if (!entry) {
            return;
        }
        if (entry.isFile) {
            entry.file(function(file) {
                uploadFile = file;
                $("#check").attr("disabled", false);
                /*显示文件名*/
                show(file);
            })
        } else {
            /*文件夹*/
            alert("检测到当前拖拽的是文件夹，请拖拽文件上传")
        }
    }
```

```
    }
    function handleDragOver(evt) {
        evt.stopPropagation();
        evt.preventDefault();
        evt.dataTransfer.dropEffect = 'copy'; // Explicitly show this is a copy.
    }
```

3. 两种上传文件的方式将上传的文件赋值给全局变量 uploadFile,并使 Button 可用。采用 ajax 将 formData 数据上传给服务器,返回生成报告的 url，通过 location.href 进行跳转。

```
    <input type="button" id="check" onclick="formSubmit()" class="btn btn-lg
btn-warning btn-block" disabled value="Generate Report"></input>
/*button 按钮*/
    function formSubmit() {
        updateCookie()
        upload();
        $("#check").val("loading...")
        $("#check").attr("disabled",true)
    }
    /*ajax 上传 formData*/
    function upload() {
        /*生成 formData*/
        var formData = new FormData();
        formData.append("file", uploadFile, uploadFile.name)
        /*ajax 异步上传*/
        var xhr = new XMLHttpRequest();
        xhr.open('post', '/healthCheckTool/HealthCheckServlet', true);
        xhr.onload = function(data) {
            if (xhr.status == 200) {
                /*返回 url 地址，跳转*/
                if(xhr.responseText == "error"){
                    $("#check").val("Generate Report")
                    $("#check").attr("disabled",true)
                    alert(xhr.responseText)
                }else{
                    location.href=xhr.responseText
                }
            } else {
                alert("error!")
            }
        }
        xhr.send(formData);
    }
```

4. Servlet 获取文件。通过 org.apache.commons.fileupload.servlet.ServletFileUpload 对 request 进行处理，生成 FileItem 的 List。将文件保存在预设的路径下。

```
List<FileItem> formItems = upload.parseRequest(request);
    System.out.println("List length:" + formItems.size());
    if (formItems != null && formItems.size() > 0) {
        // 迭代表单数据
        for (FileItem item : formItems) {
            // 处理不在表单中的字段
            if (!item.isFormField()) {
                String fileName = new File(item.getName()).getName();
                System.out.println("fileName:"+fileName);
                request.setAttribute("fileName", fileName);
                String filePath = uploadPath + File.separator + fileName;
                File storeFile = new File(filePath);
                // 在控制台输出文件的上传路径
                System.out.println(filePath);
                // 保存文件到硬盘
                item.write(storeFile);
                request.setAttribute("message", "文件上传成功!");
            }else {
                if(item.getFieldName().equals("drop")) {
                    request.setAttribute("drop", true);
                }
            }
        }
    }
```

5.2 调用 Python

1. 首先用字符串拼接上传文件的路径 uploadPath，生成报告的存储位置 generatePath 和 python 的执行路径 execPath。

```
String basePath = getServletContext().getRealPath("/") + File.separator + "MyHCA";
String execPath = basePath +File.separator+"HealthCheck";
String uploadPath = basePath + File.separator + "uploadFiles" + File.separator + req.getAttribute("fileName");
String generatePath = basePath +File.separator+"HealthCheck" + File.separator + "generatedFiles"+File.separator;
```

2. 然后拼接执行的命令 cmd，通过 Runtime.getRuntime().exec()在 execPath 下执行 cmd 命令。

```
String cmd = "python MyHCA.py "+ uploadPath+" "+ generatePath;
Process proc = Runtime.getRuntime().exec(cmd,null,new File(execPath));
```

3. 如果执行成功，将生成报告的 url 返回。

```
String htmlFilePath ="/healthCheckTool/MyHCA/HealthCheck/generatedFiles/" + reportFile + "/" + htmlFileName;
resp.getWriter().print(htmlFilePath);
```

4. 进行性能对比时，先对两个文件进行 healthCheck 操作，生成 rawData 的路径 filePath1 和 filePath2。

```
String filePath1 = healthCheck(firstName,basePath);
```

```
String filePath2 = healthCheck(secondName,basePath);
```

5. 然后拼接出生成文件的路径 generatedPath、执行的命令 cmd 和执行的路径 execPath。最后采用 Runtime.getRuntime().exec()在 execPath 路径下执行命令 cmd。

```
String generatedPath =
generateFile+"ReportFile_compare_"+firstName+"_"+secondName;
String cmd = "python compare.py "+ filePath1+" "+filePath2+" "+generatedPath;
String execPath =
getServletContext().getRealPath("/")+File.separator+"MyHCA"+File.separator+"Compare
"+File.separator;
Process pr = Runtime.getRuntime().exec(cmd,null, new File(execPath));
```

6. 如果执行成功，将生成报告的 url 返回。

```
String htmlFilePath
="/healthCheckTool/MyHCA/Compare/generatedFiles/ReportFile_compare_"+firstName+"_"+
secondName+"/ReportFile_compare_sorted.html";
response.getWriter().print(htmlFilePath);
```

6. 存在问题

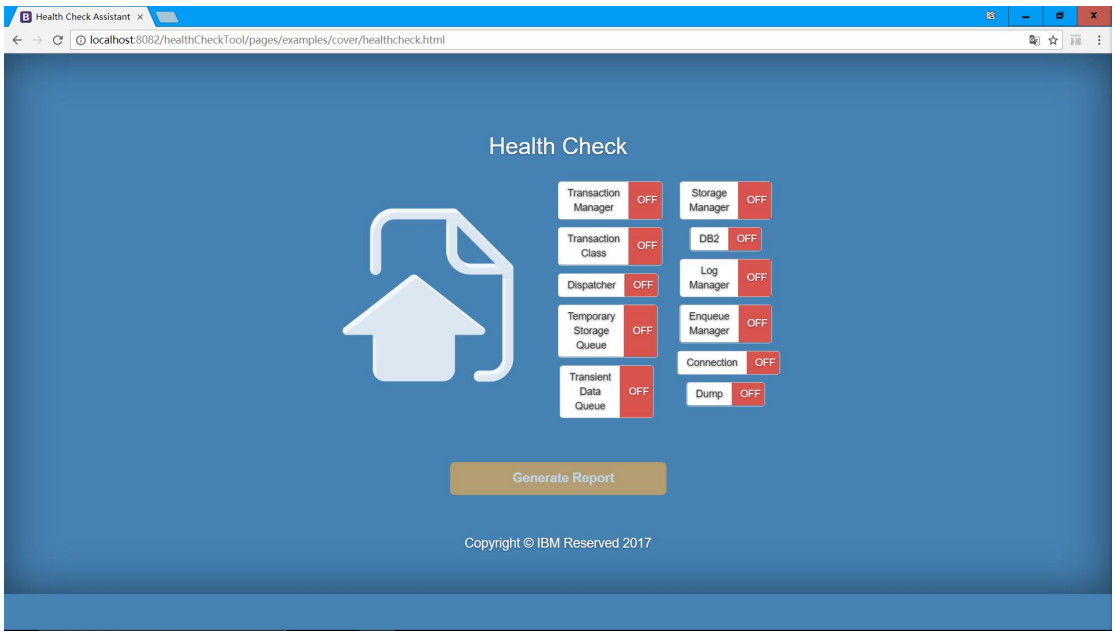


图 6.1 HealthCheck 界面

如图 6.1 所示，bootstrap-switch 渲染的 checkbox 存在宽度不一致的问题，目前尚未找到解决方案。