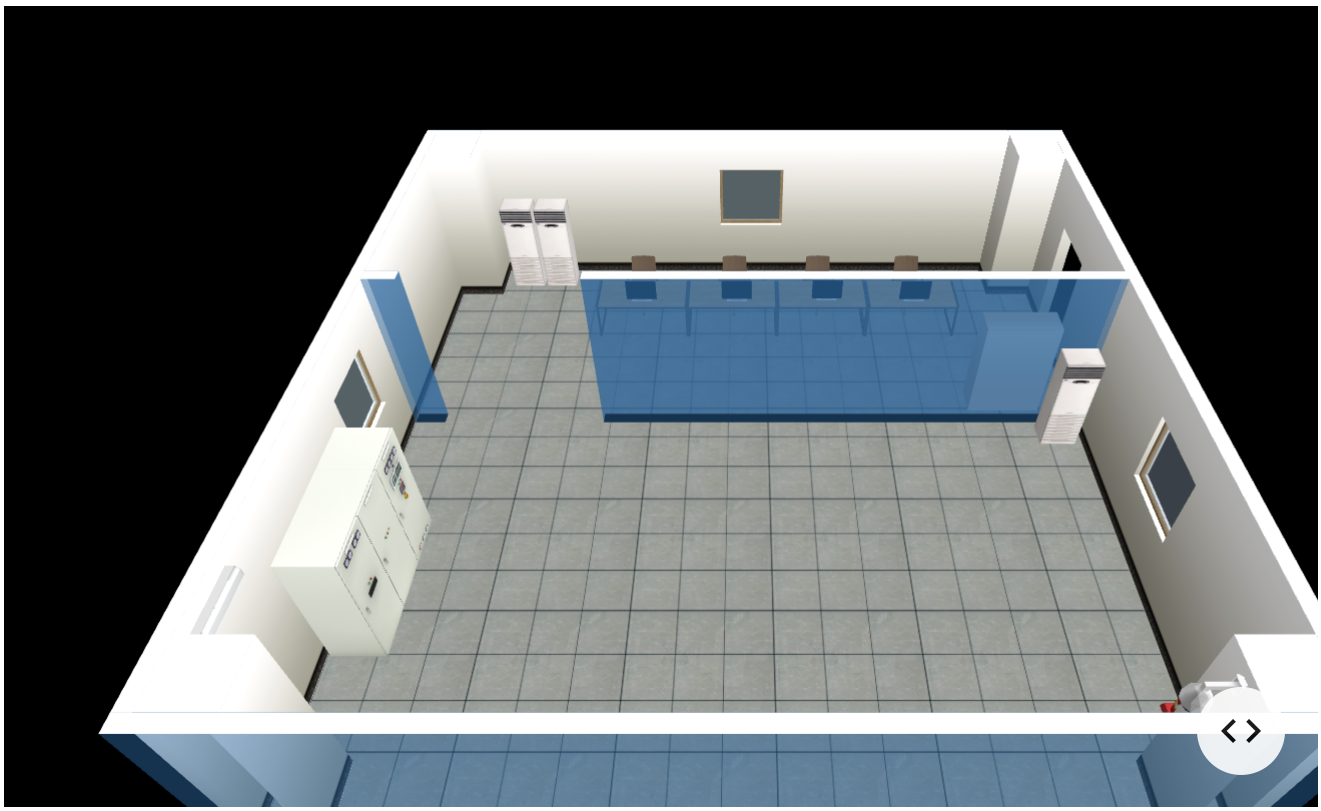
Sbm 模型

loadSbm

加载 `sbm` 模型。

样例：



定义：

```ts
interface SbmInfo extends BaseObject3DInfo, ObjectEvents<Sbm> {
  url: string;
}

function loadSbm(
  sbmInfo: SbmInfo,
  onProgress?: ModelLoadingProgressCallback
): Promise<Sbm>;
```

用法：

```js
ssp
  .loadSbm(
    // sbmInfo
    {
      id: 'xx',
      name: 'xx',
      url: 'xx/x.sbm',
      level: {
        max: 1000,
        min: null,
      },
      position: { x: 1000, y: 0, z: 1000 },
      rotation: { x: 0, y: 0, z: 0 },
      scale: { x: 2, y: 2, z: 2 },
      onClick(e) {
        /**
         * 对象的独立事件触发后，默认不传播（类似 DOM 的事件冒泡）到全局事件，
         * 调用 eventPropagation 方法通知事件继续传播到全局。
         *
         * warn:
         *  在 **非箭头函数** 中参数 e 与 this 的指向都是当前模型对象，
         *  在 *箭头函数** 参数 e 依然是模型对象，但 this 指向会发生改变。
         */
        this.eventPropagation();

        console.log('模型自身的点击事件触发', this);
      },
      onDblClick: (e) => {
        /**
         * 这里模拟在 **箭头函数** 中
         */
        e.eventPropagation();

        console.log('模型自身的双击事件触发', e);
      },
      userData: {},
    },
    // onProgress
    ({ loaded, total }) => {
      console.log(
```

```
    }
  )
  .then((sbm) => console.log(sbm))
  .catch((err) => console.error(err));
```

参数:

sbmInfo

- **描述:** 实例 `Sbm` 对象所需信息
- **类型:** SbmInfo
- **必填:** ✓

SbmInfo

属性	描述	类型	必填	默认值
id	唯一ID	string | number	✓	
name	名称	string	✗	
url	资源路径	string	✓	
level	显示层级范围	Level	✗	{ max: null, min: null }
position	位置坐标	Position	✗	{ x: 0, y: 0, z: 0 }
rotation	旋转弧度	Rotation	✗	{ x: 0, y: 0, z: 0 }
scale	缩放比例	Scale	✗	{ x: 1, y: 1, z: 1 }
userData	用户数据	any	✗	{}
onHover	鼠标悬浮事件	Function	✗	null
onUnHover	鼠标悬浮后离开事件	Function	✗	null
onClick	左键单击事件	Function	✗	null

onDblClick	左键双击事件	Function	✗	null
onRightClick	右键单击事件	Function	✗	null
onLoad	加载完成事件	Function	✗	null

onProgress

- **描述:** 模型加载时进度回调函数，回填参数包含如下字段。
- **类型:** (progress: ModelLoadingProgress) => void
- **必填:** ✗

ModelLoadingProgress

字段名	描述	类型
total	需要加载总数	number
loaded	已加载完成数量	number
timeStamp	单步消耗时长	number

parseSbm

解析 Sbm 模型

定义:

```ts
function parseSbm(
  data: ArrayBuffer,
  sbmInfo: SbmInfo,
  onProgress?: ModelLoadingProgressCallback
): Promise<Sbm>;
```

用法:

```js
const sbm = await ssp.parseSbm(new ArrayBuffer(8), sbmInfo, () => {});
```

参数:

data

- **描述:** 模型数据
- **类型:** `ArrayBuffer`
- **必填:** ✔

sbmInfo

同上

onProgress

同上

cloneSbm

克隆 Sbm 模型

定义:

```ts
interface CloneSbmInfo extends Omit<SbmInfo, 'url'> {

}

function cloneSbm(
  model: Sbm,
  cloneSbmInfo: CloneSbmInfo,
  parent?: BaseObject3D | null
): Promise<Sbm>;
```

用法:

```
    position: {
      x: 100,
      y: 0,
      z: 0
    }
});
```

参数:

model

- **描述:** Sbm 对象
- **类型:** `Sbm`
- **必填:** ✓

sbmInfo

同 Sbminfo, 但不需要字段 `url`。

parent

- **描述:** 将 `Sbm` 克隆到的 `parent` 下
- **类型:** `Sbm`
- **必填:** ✗

getSbmById

通过 `id` 查找

定义:

```ts
function getSbmById(id: SbmInfo['id']): Sbm | null;
```

用法:

getSbmByName

通过 `name` 查找

定义:

```ts
function getSbmByName(name: string): Sbm[];
```

用法:

```js
const sbmList = ssp.getSbmByName('xxx');
```

getAllSbm

获取所有 `Sbm` 对象

定义:

```ts
function getAllSbm(): Sbm[];
```

用法:

```js
const allSbmList = ssp.getAllSbm();
```

getSbmByUserDataProperty

定义：

```ts
function getSbmByUserDataProperty(
  propNameOrFindFunc: string | UserDataPropertyFindFunc,
  value?: any
): Sbm[];
```

用法：

```js
const sbmList = ssp.getSbmByUserDataProperty('propKey',  'propVal')
// or
const sbmList = ssp.getSbmByUserDataProperty(item => item['itemPropKey'] ===
```

参数：

propNameOrFindFunc

- **描述:** `userData` 内属性名 或 `find` 函数
- **类型:** string | function
- **必填:** ✓

propValue

- **描述:** `userData` 内属性值。
- **类型:** any
- **必填:** ✗

> **find 函数使用场景**
>
> ```js
> sbm.userData = {
>   people: {
>     name: 'xiaoming',
>     age: 18,
>   },
> };
> ```

```
);
```

removeSbmById

通过 `id` 移除

定义：

```ts
function removeSbmById(id: SbmInfo['id']): boolean;
```

用法：

```js
ssp.removeSbmById('xxx');
```

loadSbmToGroup

加载 `sbm` 到一个组内。

定义：

```ts
function loadSbmToGroup(
  groupInfo: GroupInfo,
  sbmInfoList: SbmInfo[],
  onProgress?: GroupProgressCallback
): Promise<Group>;
```

用法：

```
    // groupInfo
    {
      id: 'firstSbmGroup',
      name: 'name_firstSbmGroup',
      // ...
    },
    // sbmInfoList
    [sbmInfo1, sbmInfo2, sbmInfo3],
    // onProgress
    ({ current, modelTotal, loadingModelIndex }) => {
      console.log(
        '模型加载进度: ',
        Math.round(
          loadingModelIndex / modelTotal +
            (current.loaded / current.total) * (1 / modelTotal) * 10000
        ) /
          100 +
          '%'
      );
    }
  )
  .then((group) => console.log(group));
```

参数

groupInfo

- **描述:** 实例组对象所需信息
- **类型:** GroupInfo
- **必填:** ✓

GroupInfo

属性	描述	类型	必填	默认值
id	组唯一ID	string | number	✓	
name	组名称	string	✗	
level	显示层级范围	Level	✗	{ max: null, min: null }

position	位置坐标	Position	✗	{ x: 0, y: 0, z: 0 }
rotation	旋转弧度	Rotation	✗	{ x: 0, y: 0, z: 0 }
scale	缩放比例	Scale	✗	{ x: 1, y: 1, z: 1 }
userData	用户数据	any	✗	{}

sbmInfoList

- **描述:** `sbmInfo` 集合
- **类型:** sbminfo[]
- **必填:** ✓

onProgress

- **描述:** 模型加载到组内时进度回调函数，回填参数包含如下字段。
- **类型:** (groupProgress: GroupProgress) => void
- **必填:** ✗

GroupProgress

字段名	描述	类型
modelTotal	需要加载的模型总数（sbmInfoList长度）	number
loadingModelIndex	当前正在加载的模型索引	number
current	当前在在加载模型的详细进度	ModelLoadingProgress

createGroupForSbm

为 `sbm` 提前创建一个空组。

> **使用场景**
>
> 与 `loadSbmToGroup` 不同，有些时候可能你还没有具体的 `sbmInfo` 数据，但你想提前创建一个批量管理的空组，当有数据时再使用 addSbmForGroup 插入。

定义：

```ts
function createGroupForSbm(groupInfo: GroupInfo): Group;
```

用法：

```js
ssp.createGroupForSbm({
  id: 'firstSbmGroup',
  name: 'name_firstSbmGroup',
  // ...
});
```

参数

groupInfo

- **描述:** 实例组对象所需信息
- **类型:** GroupInfo
- **必填:** ✓

addSbmForGroup

向一个已经存在的组内添加 `sbm` 对象。

定义：

```ts
function addSbmForGroup(
  groupId: GroupInfo['id'],
  sbmInfoList: SbmInfo[],
  onProgress?: GroupProgressCallback
): Promise<Group | null>;
```

用法:

```js
ssp
  .addSbmForGroup(
    // groupId
    'firstSbmGroup',
    // sbmInfoList
    [sbmInfo4, sbmInfo5],
    // onProgress
    (progress) => console.log('进度信息: ', progress)
  )
  .then((group) => console.log(group));
```

参数

groupId

- **描述:** 组 `id`
- **类型:** groupId['id']
- **必填:** ✓

sbmInfoList

- **类型:** sbminfo[]
- **描述:** `sbmInfo` 集合
- **必填:** ✓

onProgress

- **描述:** 模型加载到组内时进度回调函数，回填参数包含如下字段。
- **类型:** GroupProgressCallback
- **必填:** ✗

createSbmGroupFromXml

创建 Sbm 组，从 xml 文件资源

定义:

```ts
function createSbmGroupFromXml(
  groupInfo: GroupInfo,
  url: string
): Promise<Group>;
```

用法:

```js
const group = await ssp.createSbmGroupFromXml(
  // groupInfo
  {
    id: 'firstSbmGroup',
    name: 'name_firstSbmGroup',
    // ...
  },
  // url
  'xxx.xml'
);
```

参数:

groupInfo

同上

url

- **描述:** xml 文件的 `url` 地址
- **类型:** `string`
- **必填:** ✓

getSbmGroupById

定义：

```ts
function getSbmGroupById(id: GroupInfo['id']): Group | null;
```

用法：

```js
const group = ssp.getSbmGroupById('firstSbmGroup');
```

getSbmGroupByName

通过 `name` 查找 `sbm` 组

定义：

```ts
function getSbmGroupByName(name: string): Group[];
```

用法：

```js
const groupList = ssp.getSbmGroupByName('name_firstSbmGroup');
```

removeSbmGroupById

通过 `id` 移除 `sbm` 组

定义：

用法：

```js
const isRemoveSuccess = ssp.removeSbmGroupById('firstSbmGroup');
```

clearSbm

清除当前场景内所有 `sbm` 对象。

定义：

```ts
function clearSbm(): void;
```

用法：

```js
ssp.clearSbm();
```

getAllSbm

获取所有 Sbm 模型

定义：

```ts
function getAllSbm(): Sbm[];
```

用法：

showAllSbm

显示当前场景内所有 `sbm` 对象。

定义：

```ts
function showAllSbm(): void;
```

用法：

```js
ssp.showAllSbm();
```

hideAllSbm

隐藏当前场景内所有 `sbm` 对象。

定义：

```ts
function hideAllSbm(): void;
```

用法：

```js
ssp.hideAllSbm();
```

getSbmModelMaps

定义:

```ts
function getSbmModelMaps(): Map<string, Sbm>;
```

用法:

```js
const sbmMaps = ssp.getSbmModelMaps();
```

setSbmModelMaps

设置 Sbm 模型缓存

定义:

```ts
function setSbmModelMaps(maps: Map<string, Sbm>): void;
```

用法:

```js
const maps = new Map();

ssp.setSbmModelMaps(maps);
```

clearIdb

清空本地 indexedDB 模型缓存数据

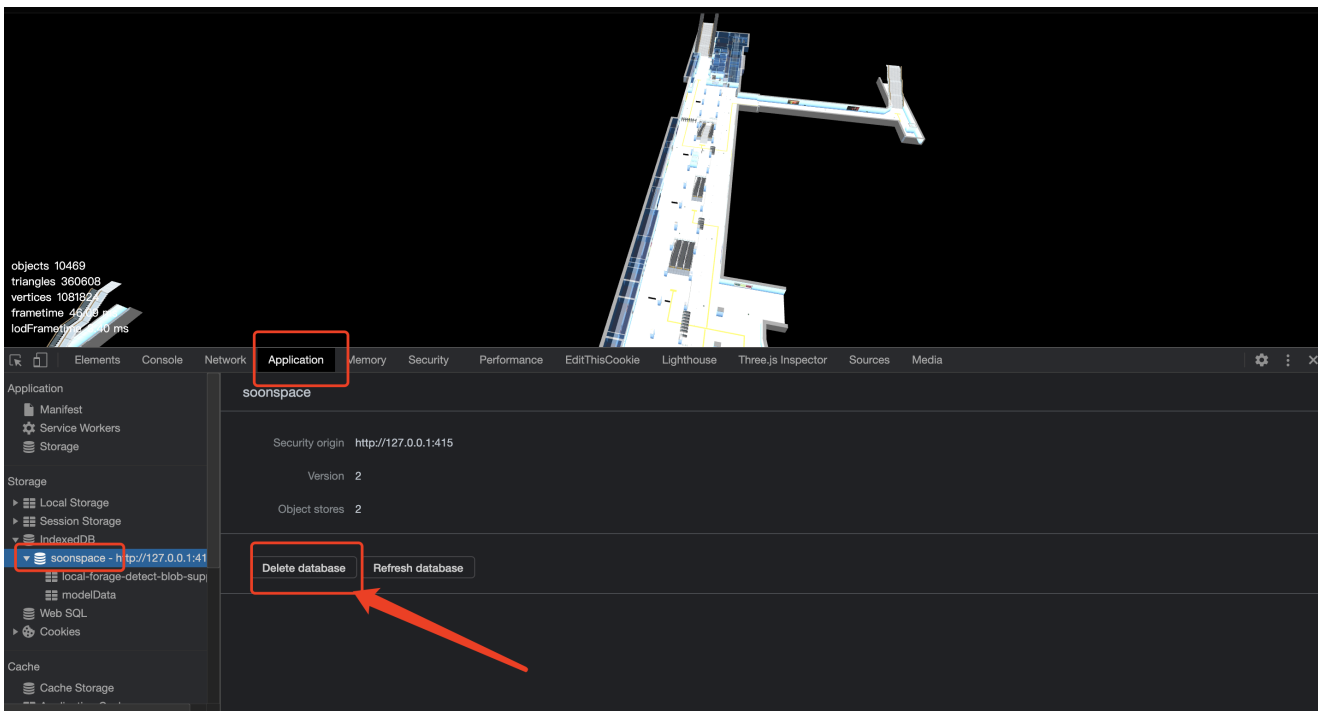
定义:

```ts
function clearIdb(): Promise<void>;
```

用法:

```js
ssp.clearIdb().then(() => {
  console.log('本地数据已清空!!!');
});
```

手动清除 indexedDB 缓存



通用模型 →