# Initial stages, initial explorations

## Or: how to find your way in an XML universe

NEH Institute “Advanced Digital Editing”
Week 1, day 1
July 11: 11am - 12.30pm

# Outcomes

- A basic understanding of XPath
- Getting acquainted with eXide
- Seeing how to navigate an XML document in eXide with XPath

# 1. TEI XML and XPath

# XML datamodel

**Ordered Hierarchy of Content Objects (OHCO)**

*“text is best represented as an ordered hierarchy of content objects (OHCO), because that is what text really is.”*

DeRose et al. (1990), “What is text, really?”

*“we have retreated from the simple OHCO thesis […] texts qua intellectual objects still seem to be composed of structures of meaning-related features and that, moreover, these structures are often hierarchical.”*

Renear, Allen H. et al (1993) “Refining our notion of what text really is: The problem of overlapping hierarchies.”

# XML datamodel

- XML is a tree
- The XML tree has a single root and many nodes
- Nodes can be of different types:
  - Document
  - Root
  - Elements
  - Attributes
  - Text
  - …
- Nodes have a relationship to each other: child, parent, ancestors, descendants…
- What you see in your editor is a serialization of the XML tree: a textual expression of the datamodel

# XML datamodel

- XML document must be well-formed
  - `<p>Some text</p>`
  - `<p>Some <l>more</p> text</l>`
  - `<div type=act>Even more text</div>`
- An XML document can be valid
- A schema allows you to specify
  - which elements can appear as the root of the XML document
  - which elements  and attributes can appear where
  - names, date types and default values for all attributes

# XPath

- XML Path Language
- A way of navigating a document that is modeled as a tree (like XML)
- An XPath expression matches a (set of) node(s) in an XML document
- An XPath expression is an “address” for the (set of) node(s), similar to a file path
- XPath is a good way to understand how your XML file is structured

# XPath

There are three components to XPath

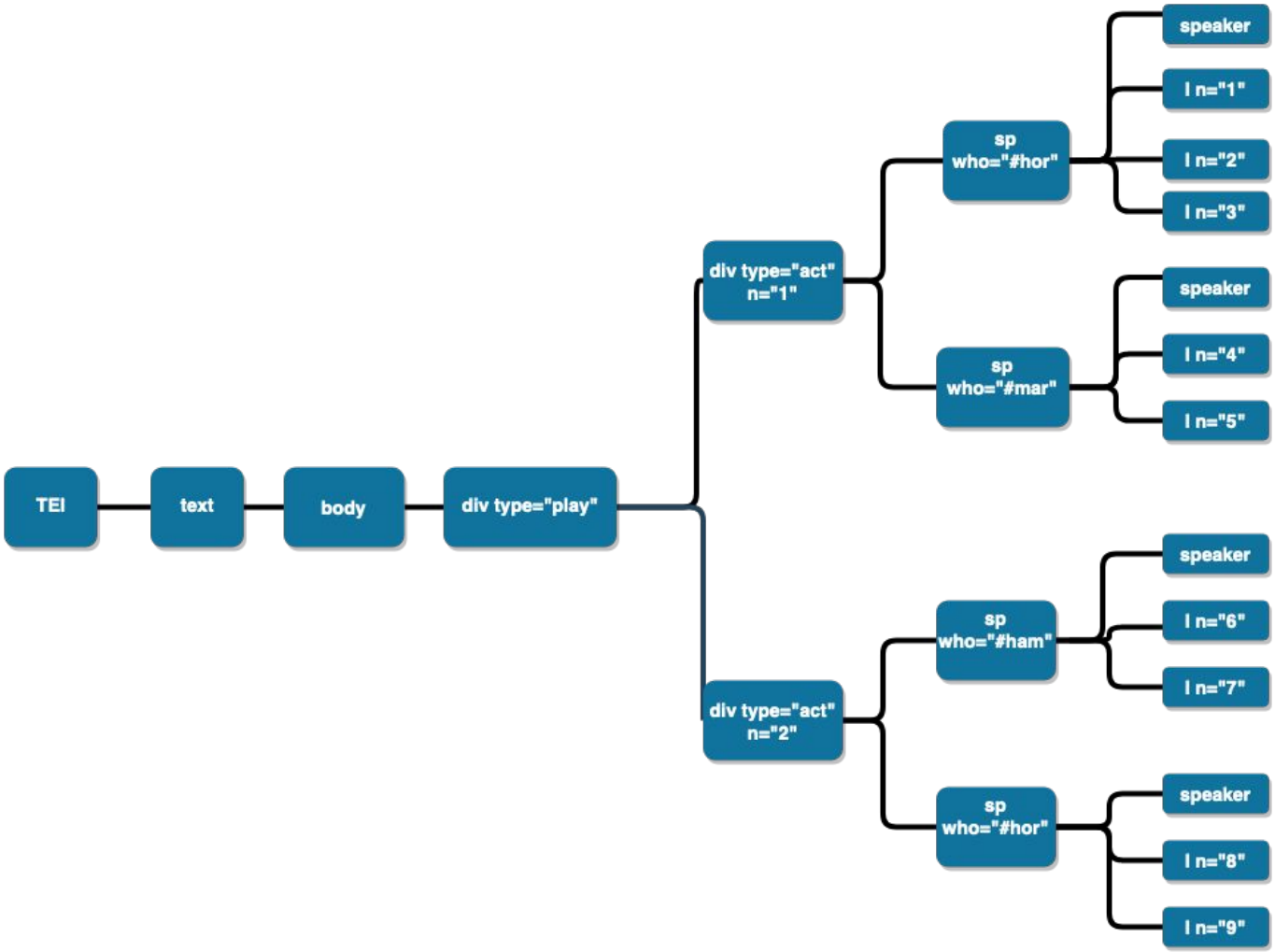
- Path expressions: navigate and traverse
- Functions: take input and produce output
- Predicates: to filter

We will focus on path expressions in this session.

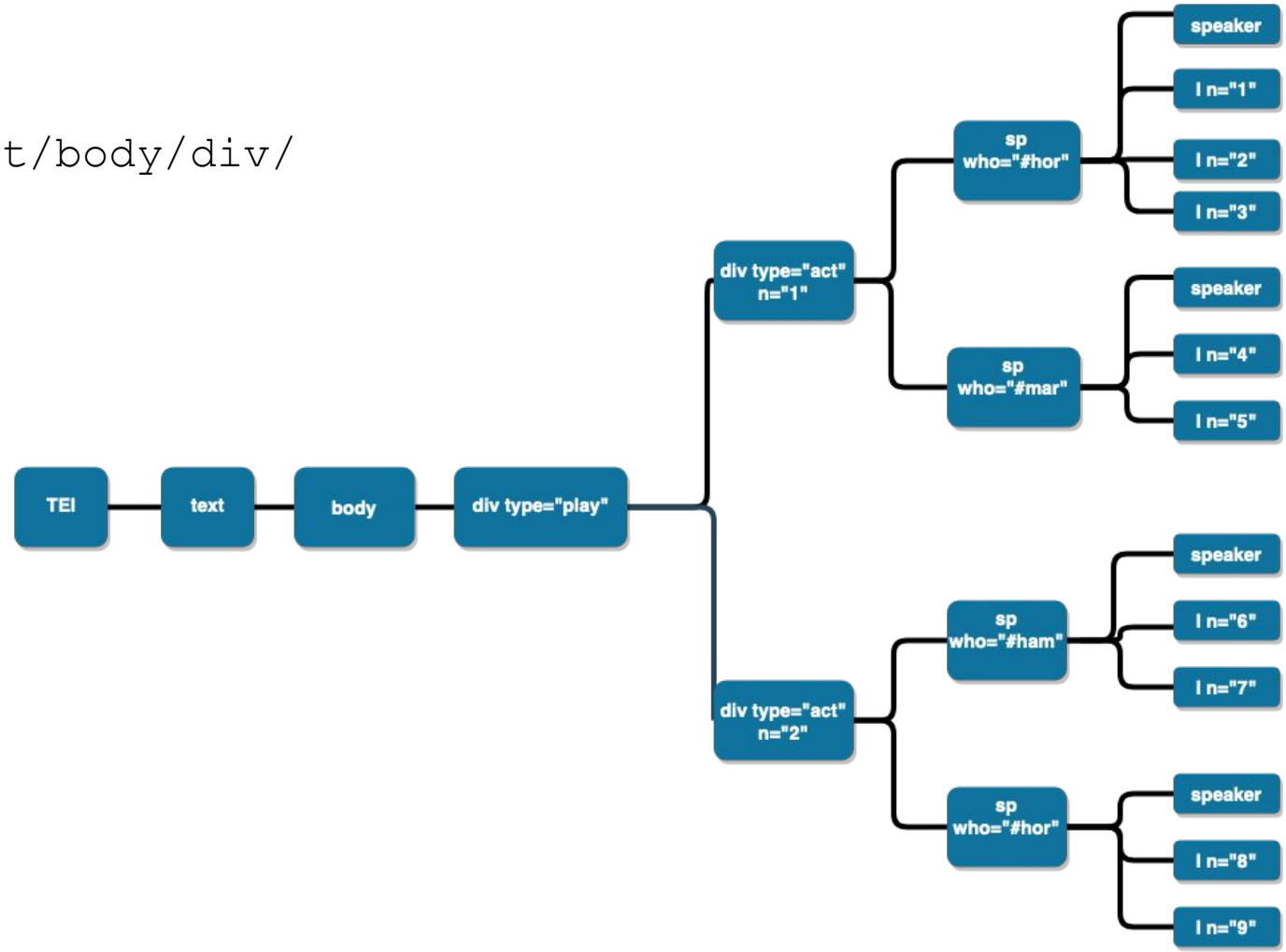
# Basic XPath syntax

| | |
|---|---|
| / | path operator, selecting a level |
| // | multi-level selector |
| * | wildcard |
| @ | attribute |
| @* | attribute wildcard |
| [...] | filter / condition |

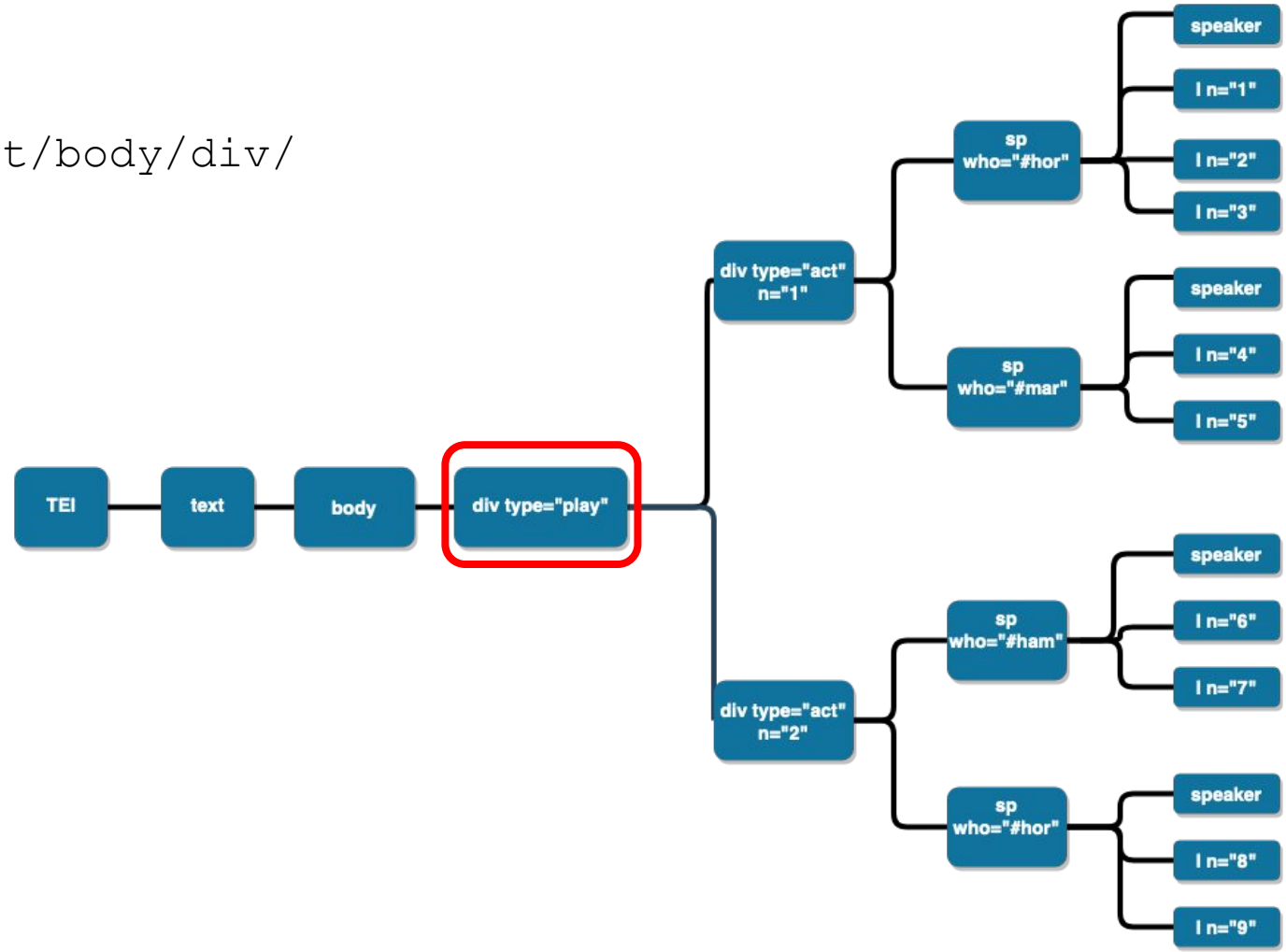
```
<TEI xmlns="http://www.tei-c.org/ns/1.0">
    <!-- TEI Header -->
    <text>
        <body>
            <div type="play">
                <div type="act" n="1">
                    <sp who="#hor">
                        <speaker>Horatio.</speaker>
                        <l n="1">What art thou that vsurp'st this time of night,</l>
                        <l n="2">Together with that Faire and Warlike forme</l>
                        <l n="3">In which the Maiesty of buried Denmarke</l>
                    </sp>
                    <sp who="#mar">
                        <speaker>Marcellus</speaker>
                        <l n="4">Peace, breake thee of:</l>
                        <l n="5">Looke where it comes againe.</l>
                    </sp>
                </div>
                <div type="act" n="2">
                        <sp who="#ham">
                        <speaker>Hamlet.</speaker>
                        <l n="6">Sir my good friend,</l>
                        <l n="7">Ile change that name with you:</l>
                    </sp>
                    <sp who="#hor">
                        <speaker>Horatio</speaker>
                        <l n="8">Season your admiration for a while</l>
                        <l n="9">With an attent eare; till I may deliuer</l>
                    </sp>
                </div>
            </div>
        </body>
    </text>
</TEI>
```


TEI
text
body
div type="play"
div type="act" n="1"
sp who="#hor"
speaker
l n="1"
l n="2"
l n="3"
sp who="#mar"
speaker
l n="4"
l n="5"
div type="act" n="2"
sp who="#ham"
speaker
l n="6"
l n="7"
sp who="#hor"
speaker
l n="8"
l n="9"

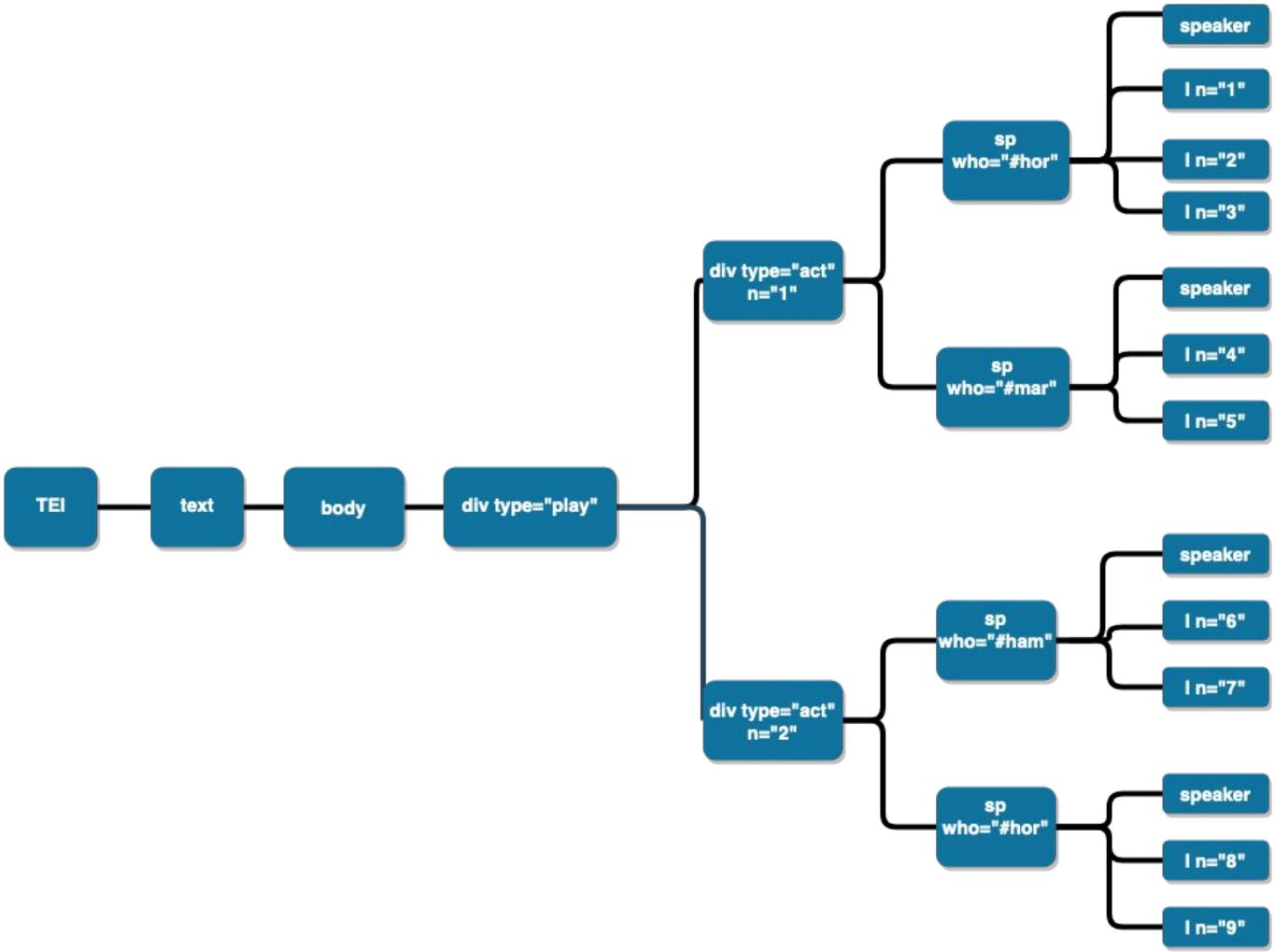
TEI/text/body/div/



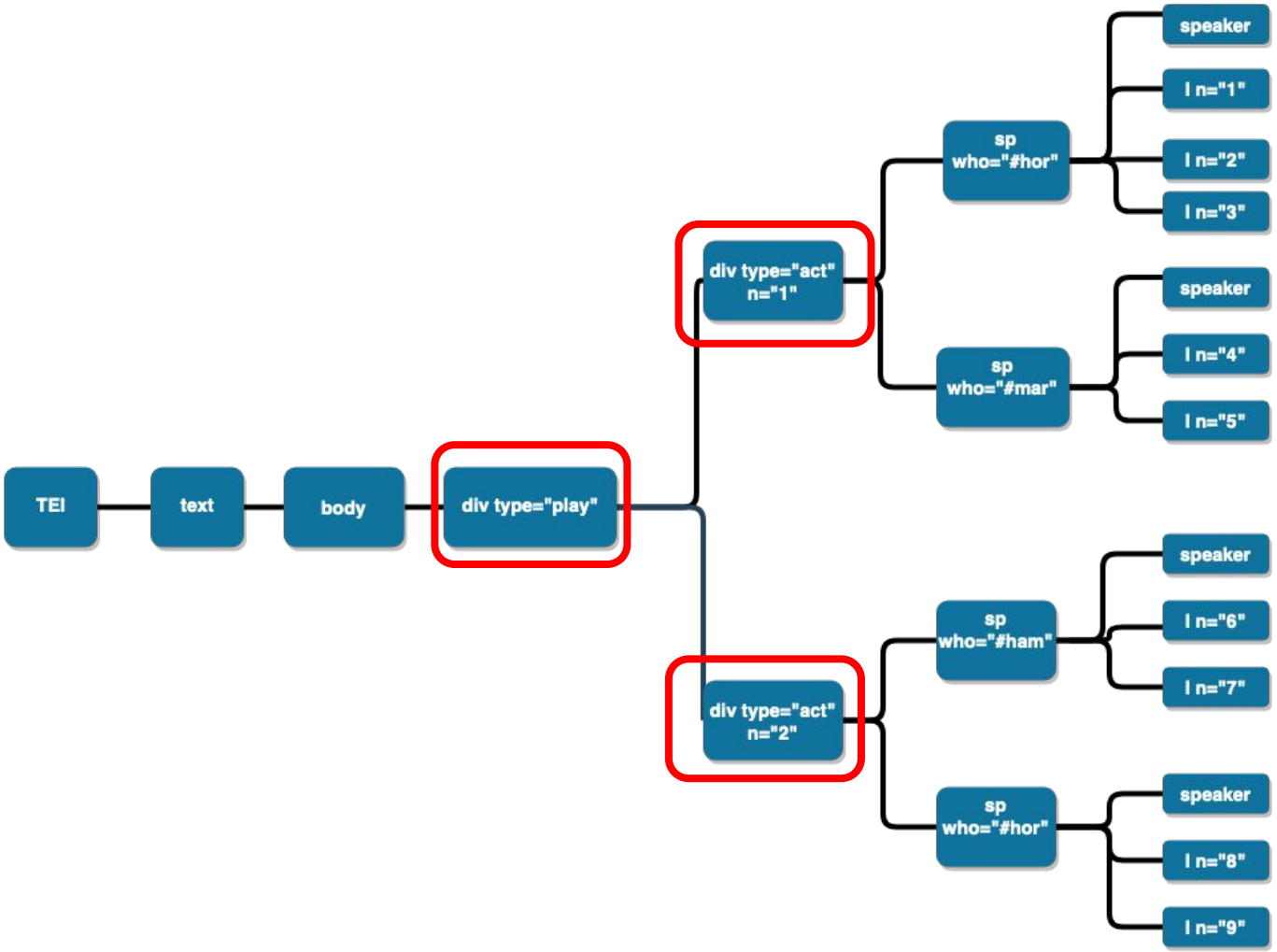
TEI/text/body/div/



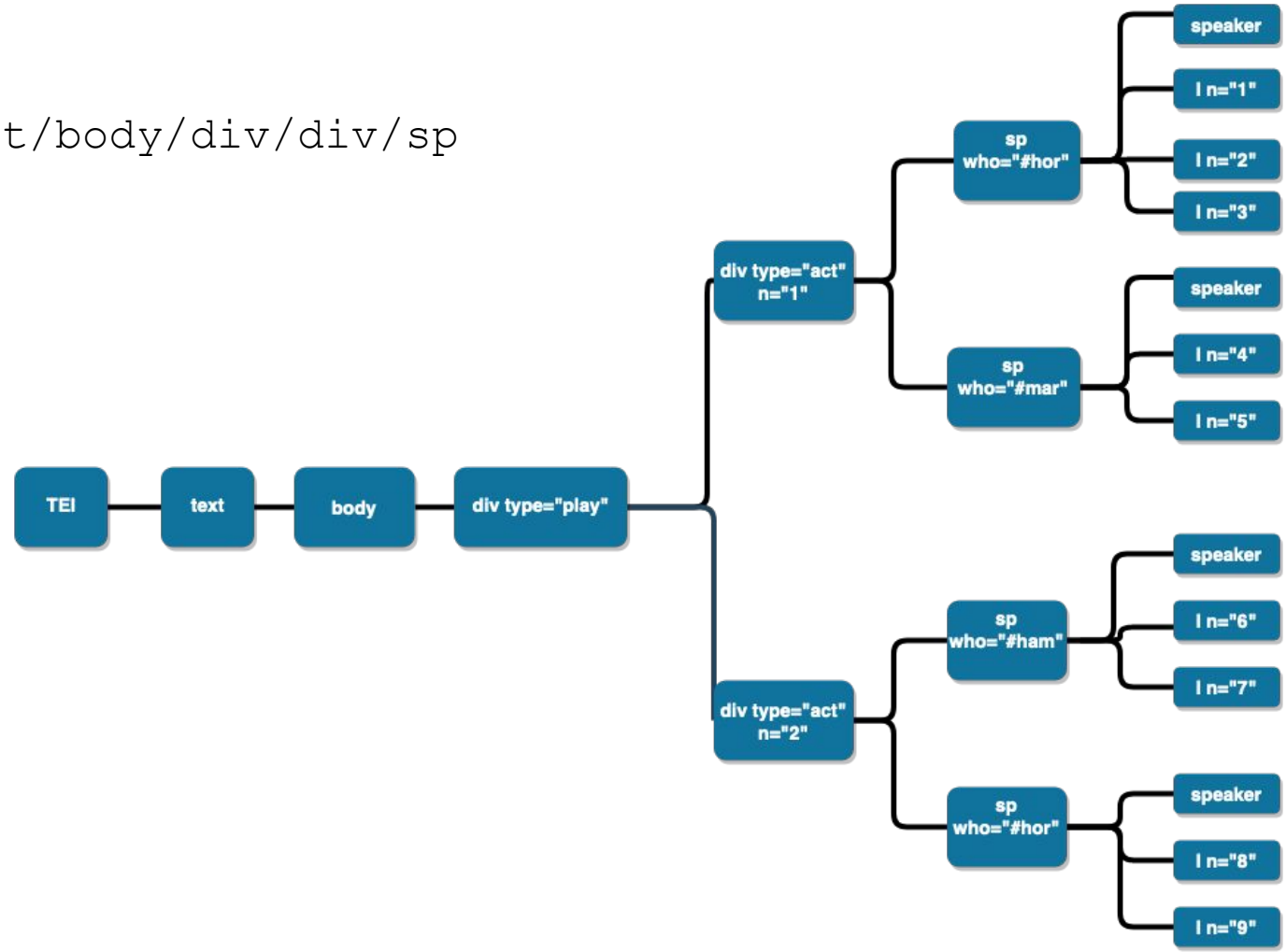
```
//div
```


TEI
text
body
div type="play"
div type="act" n="1"
sp who="#hor"
speaker
l n="1"
l n="2"
l n="3"
sp who="#mar"
speaker
l n="4"
l n="5"
div type="act" n="2"
sp who="#ham"
speaker
l n="6"
l n="7"
sp who="#hor"
speaker
l n="8"
l n="9"

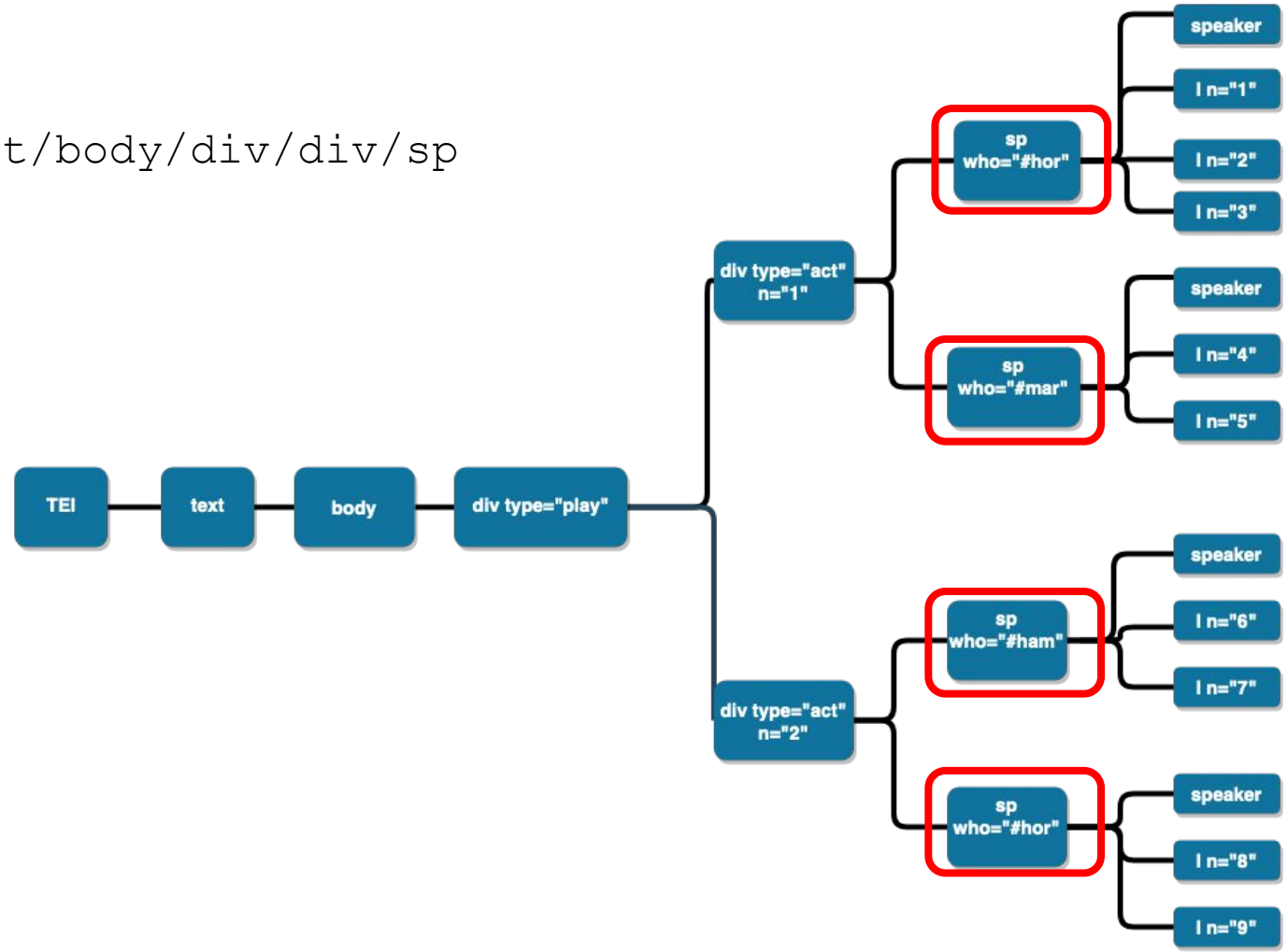
```
//div
```


TEI
text
body
div type="play"
div type="act" n="1"
sp who="#hor"
speaker
l n="1"
l n="2"
l n="3"
sp who="#mar"
speaker
l n="4"
l n="5"
div type="act" n="2"
sp who="#ham"
speaker
l n="6"
l n="7"
sp who="#hor"
speaker
l n="8"
l n="9"

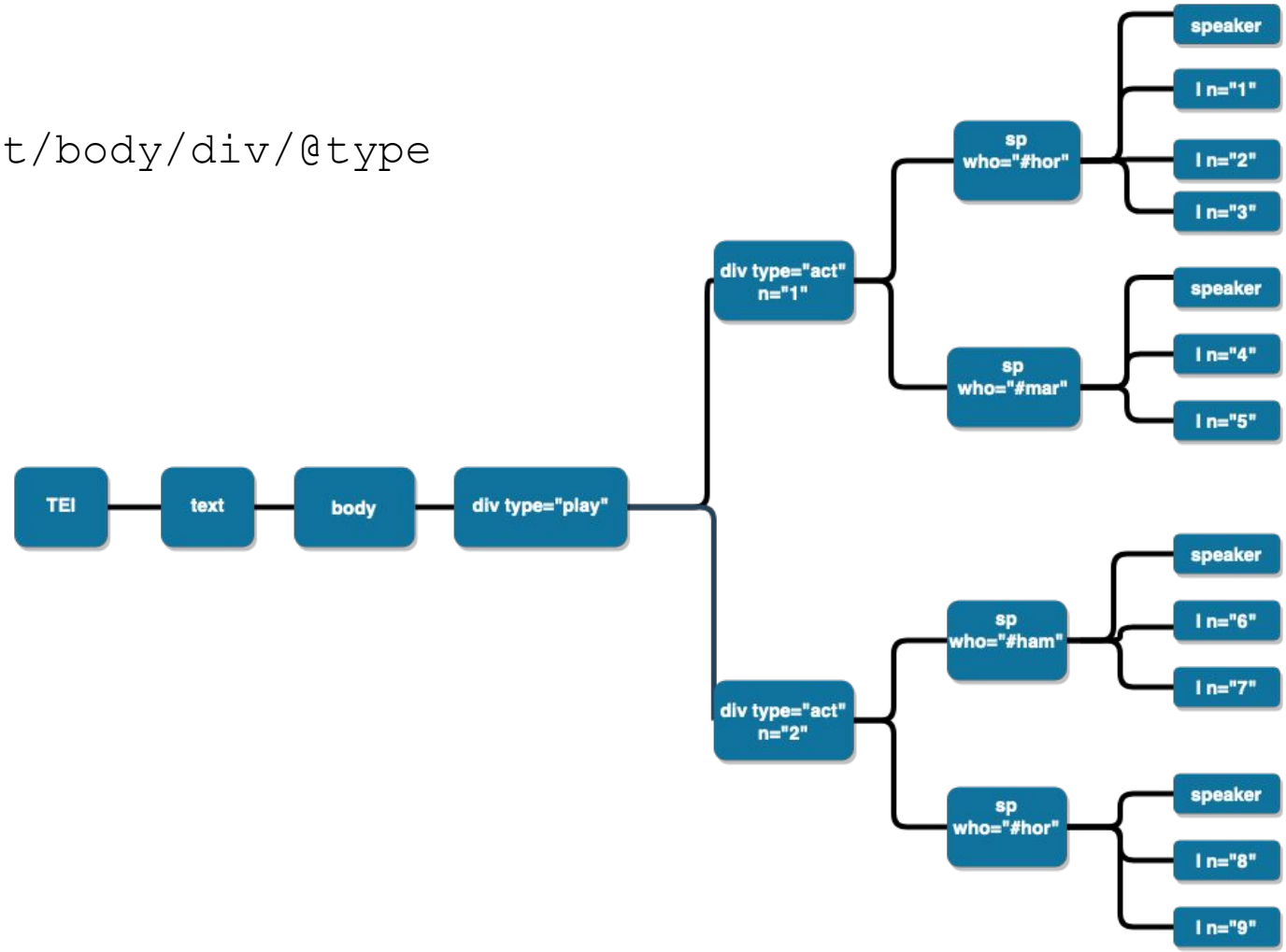
TEI/text/body/div/div/sp



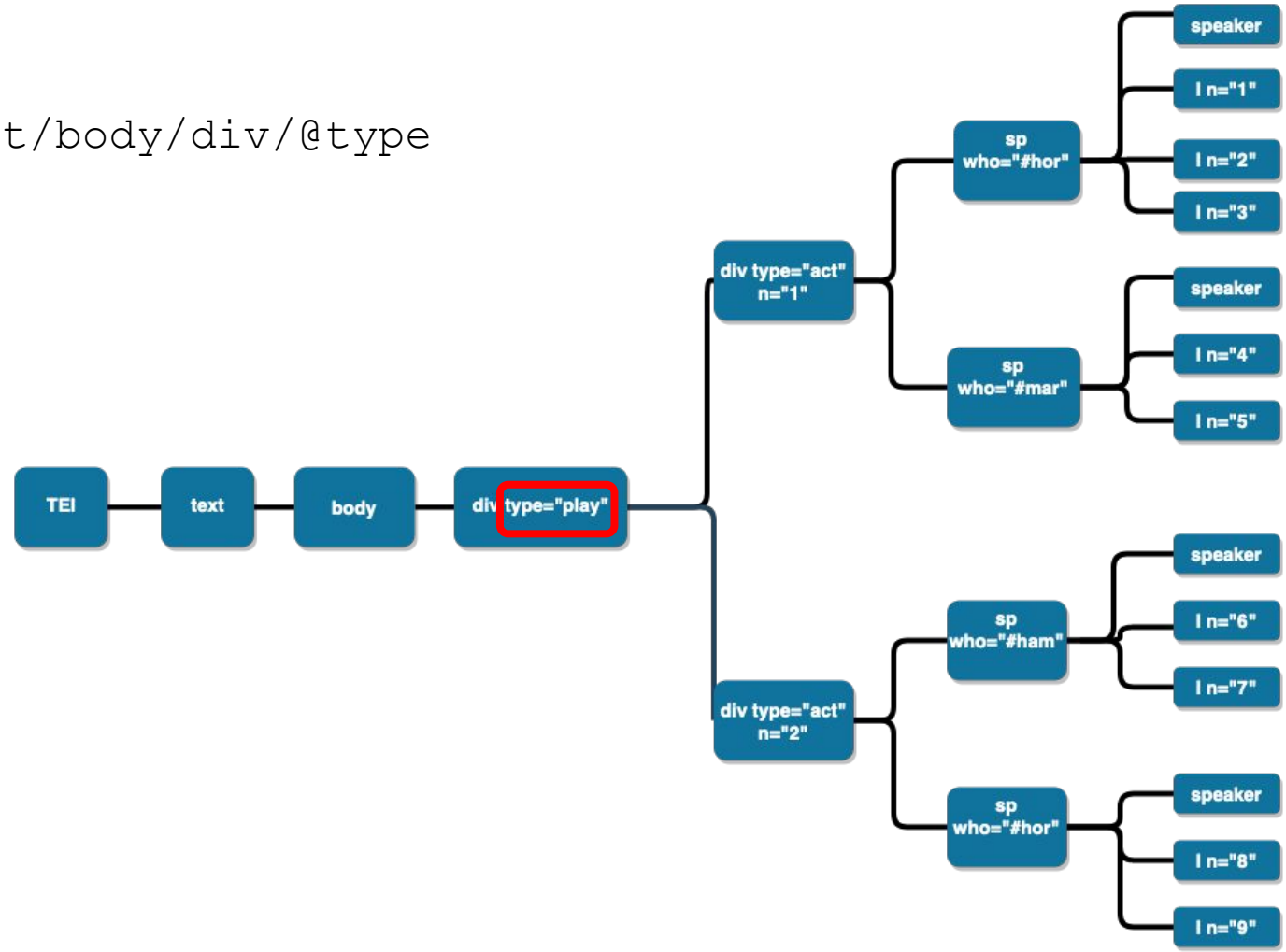
TEI/text/body/div/div/sp



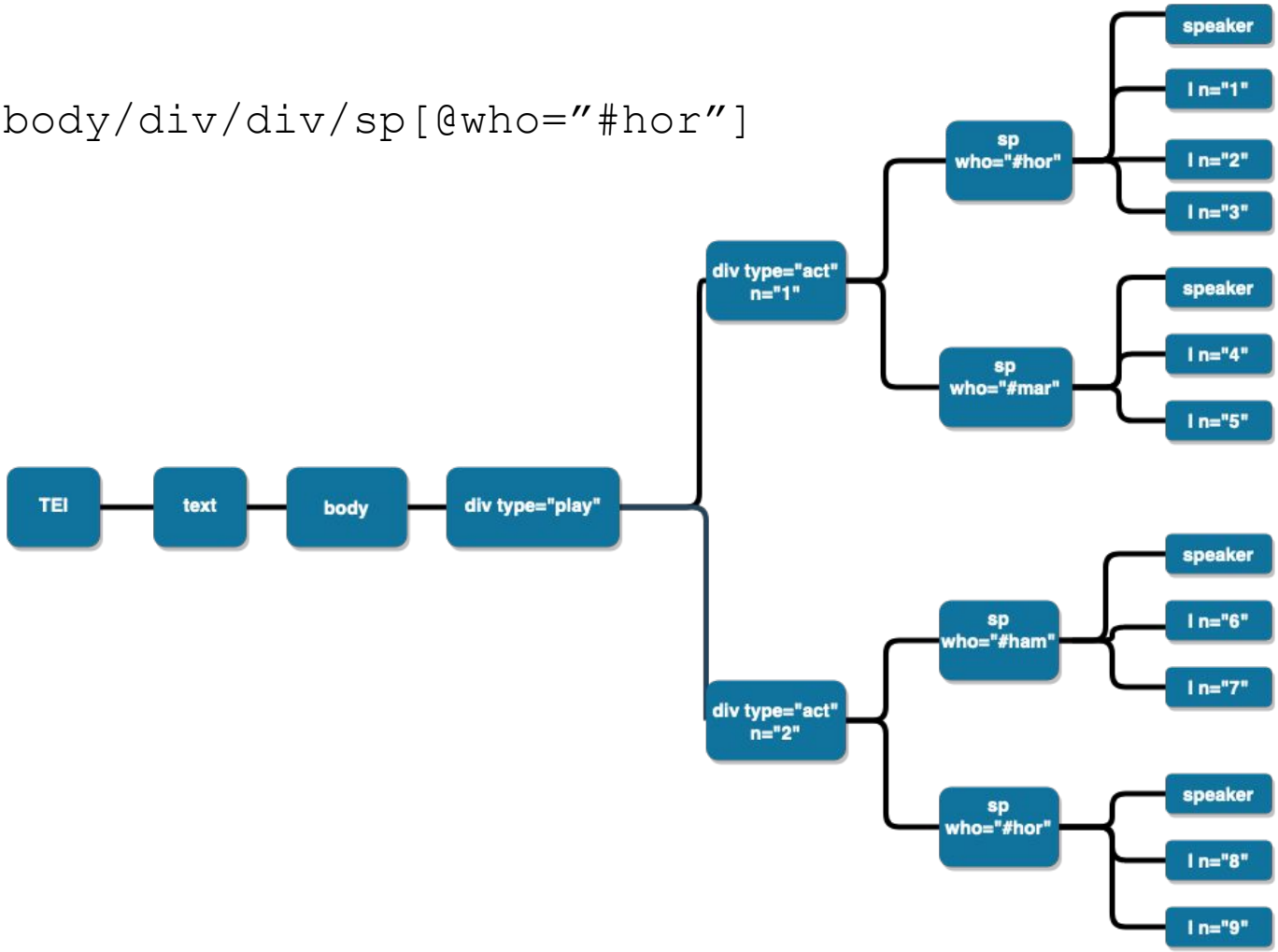
TEI/text/body/div/@type



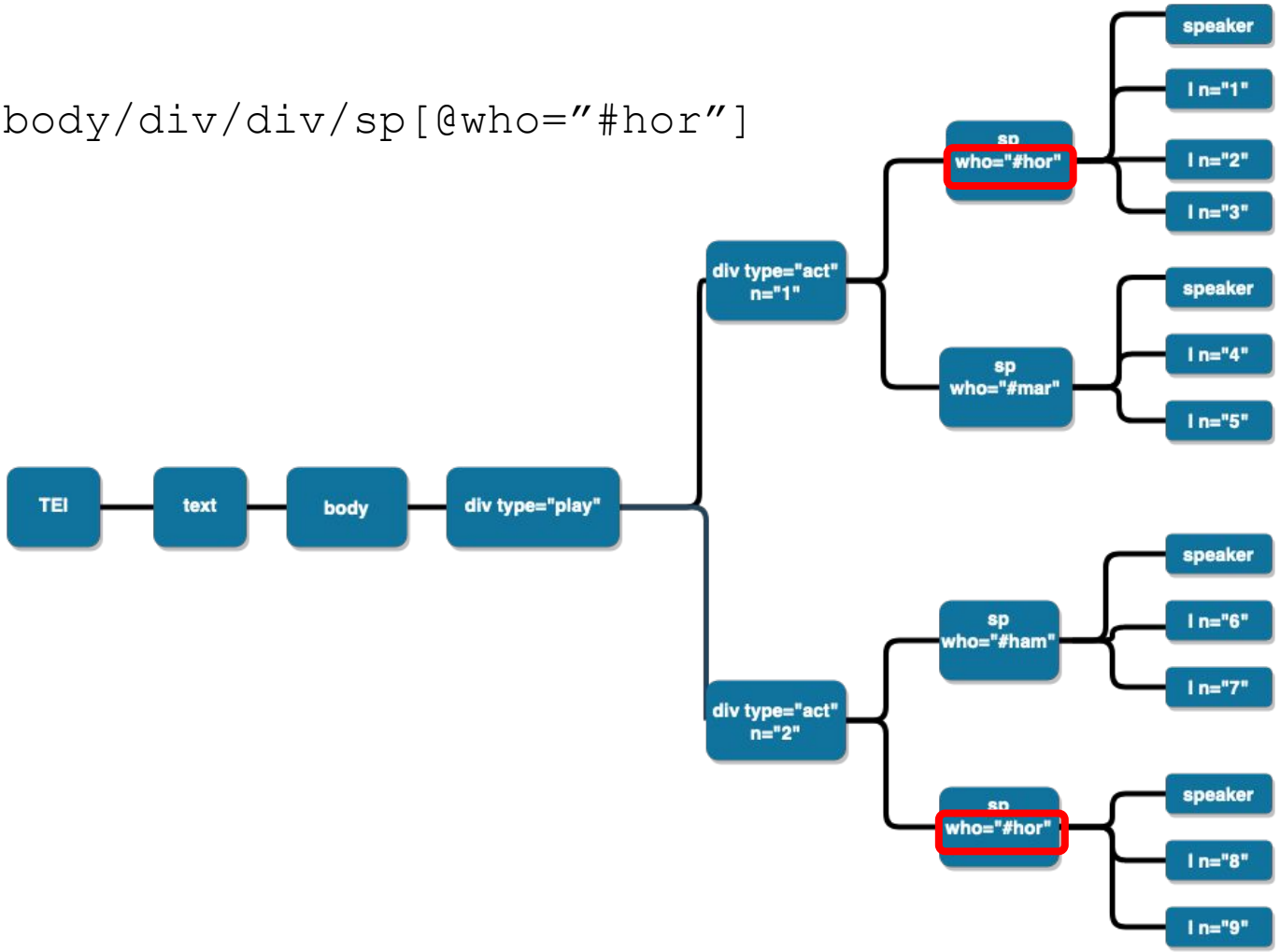
TEI/text/body/div/@type



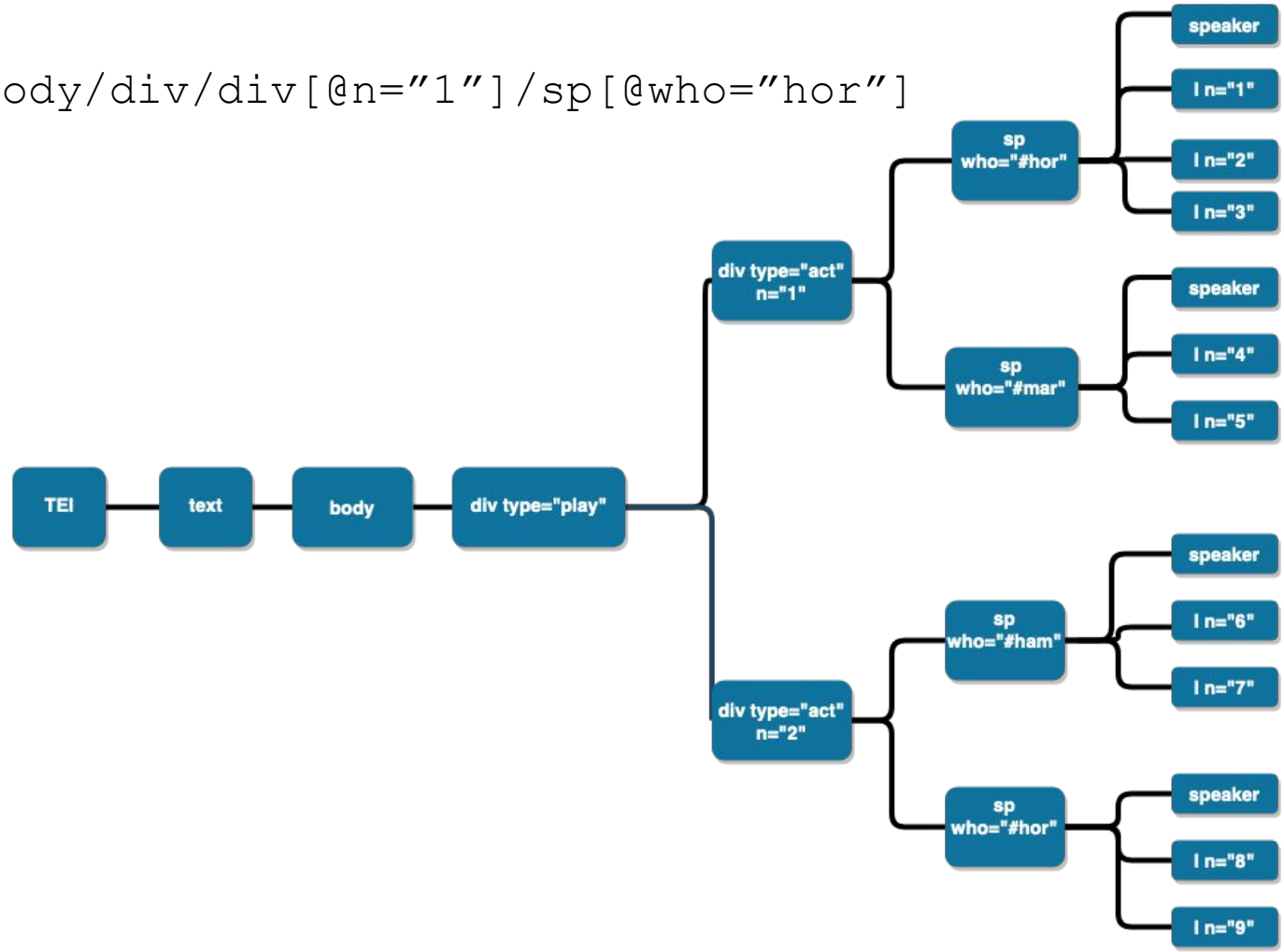
TEI/text/body/div/div/sp[@who=”#hor”]



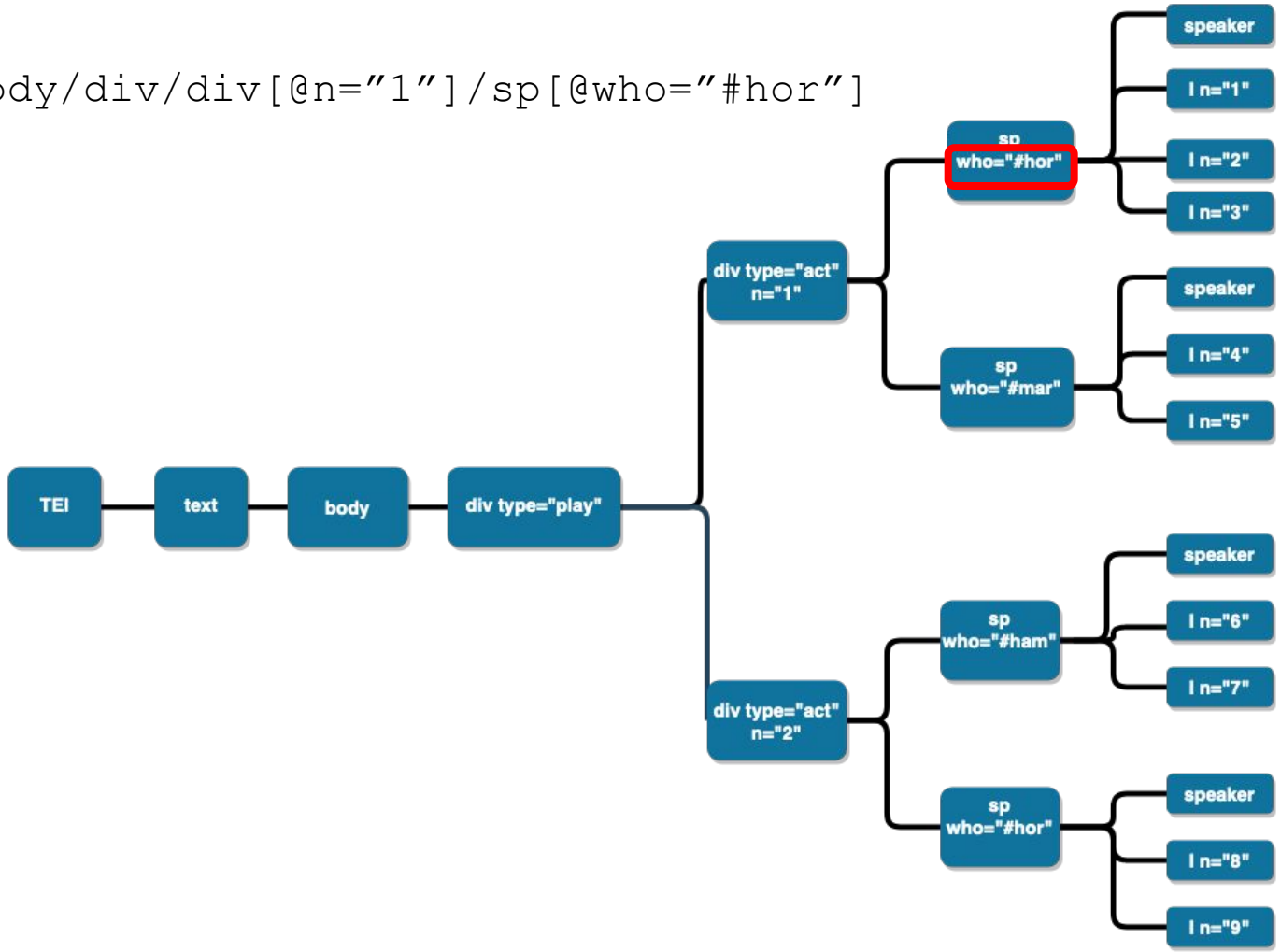
TEI/text/body/div/div/sp[@who=”#hor”]



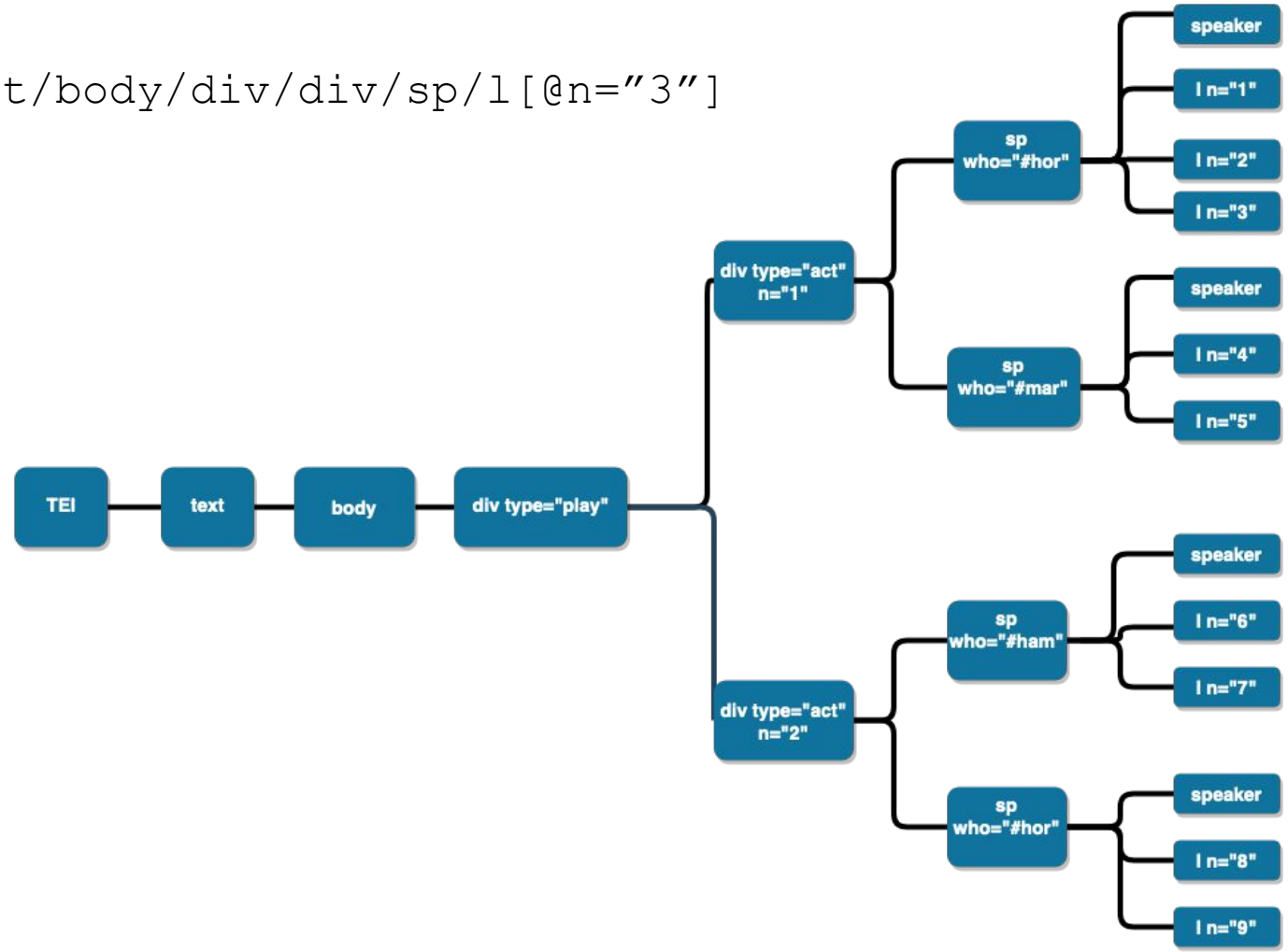
TEI/text/body/div/div[@n=”1”]/sp[@who=”hor”]



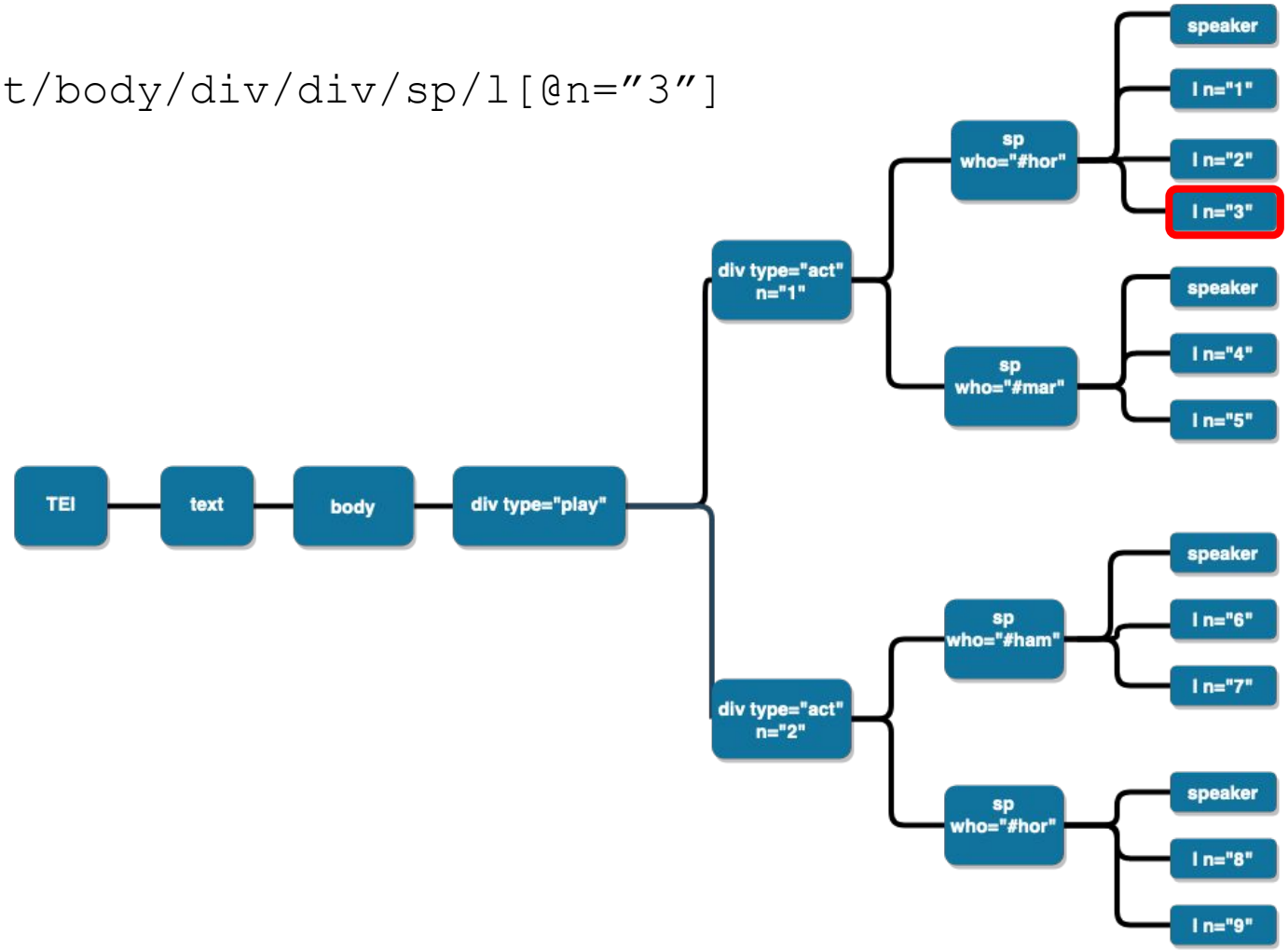
TEI/text/body/div/div[@n=”1”]/sp[@who=”#hor”]



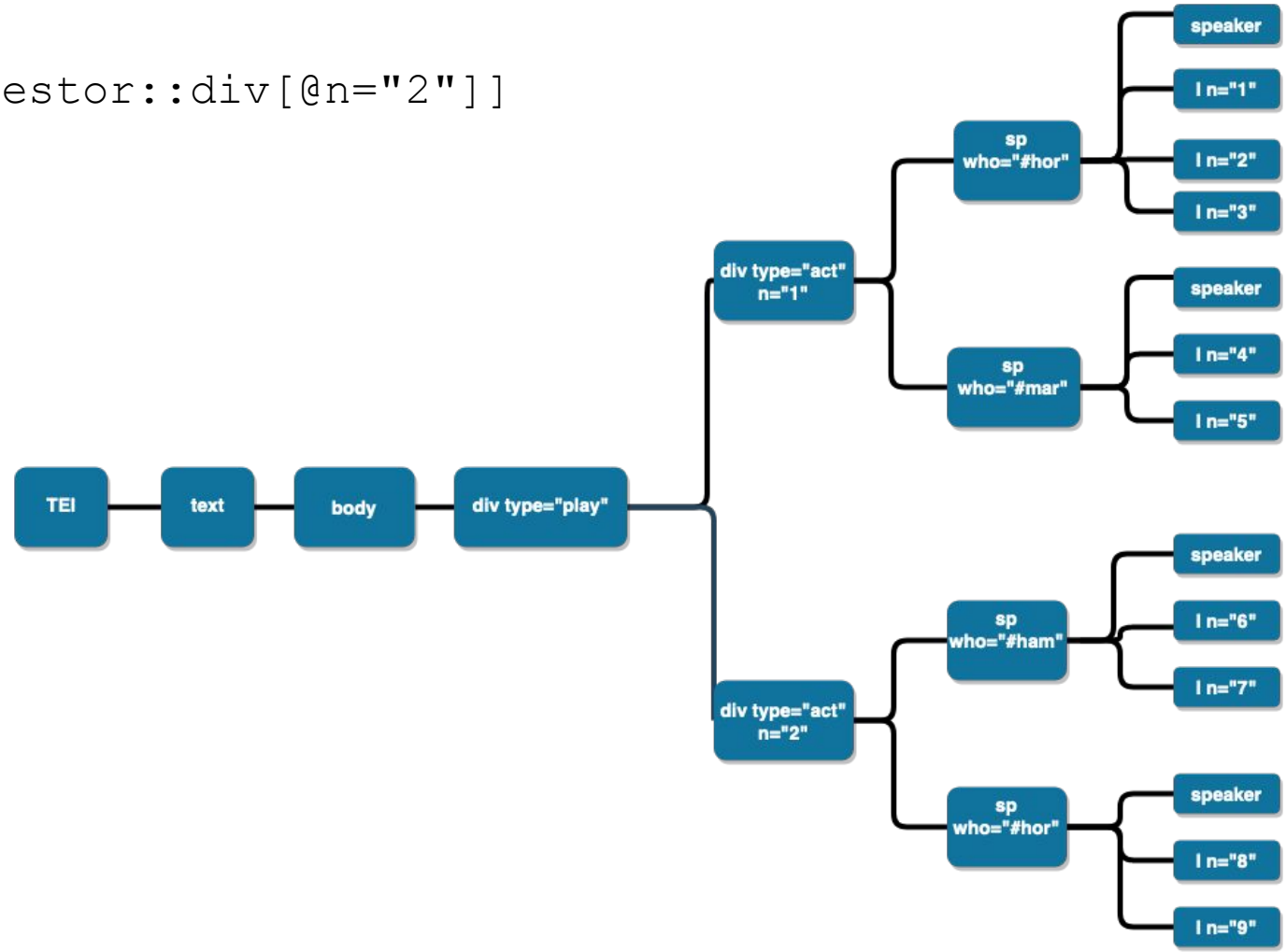
TEI/text/body/div/div/sp/l[@n=”3”]



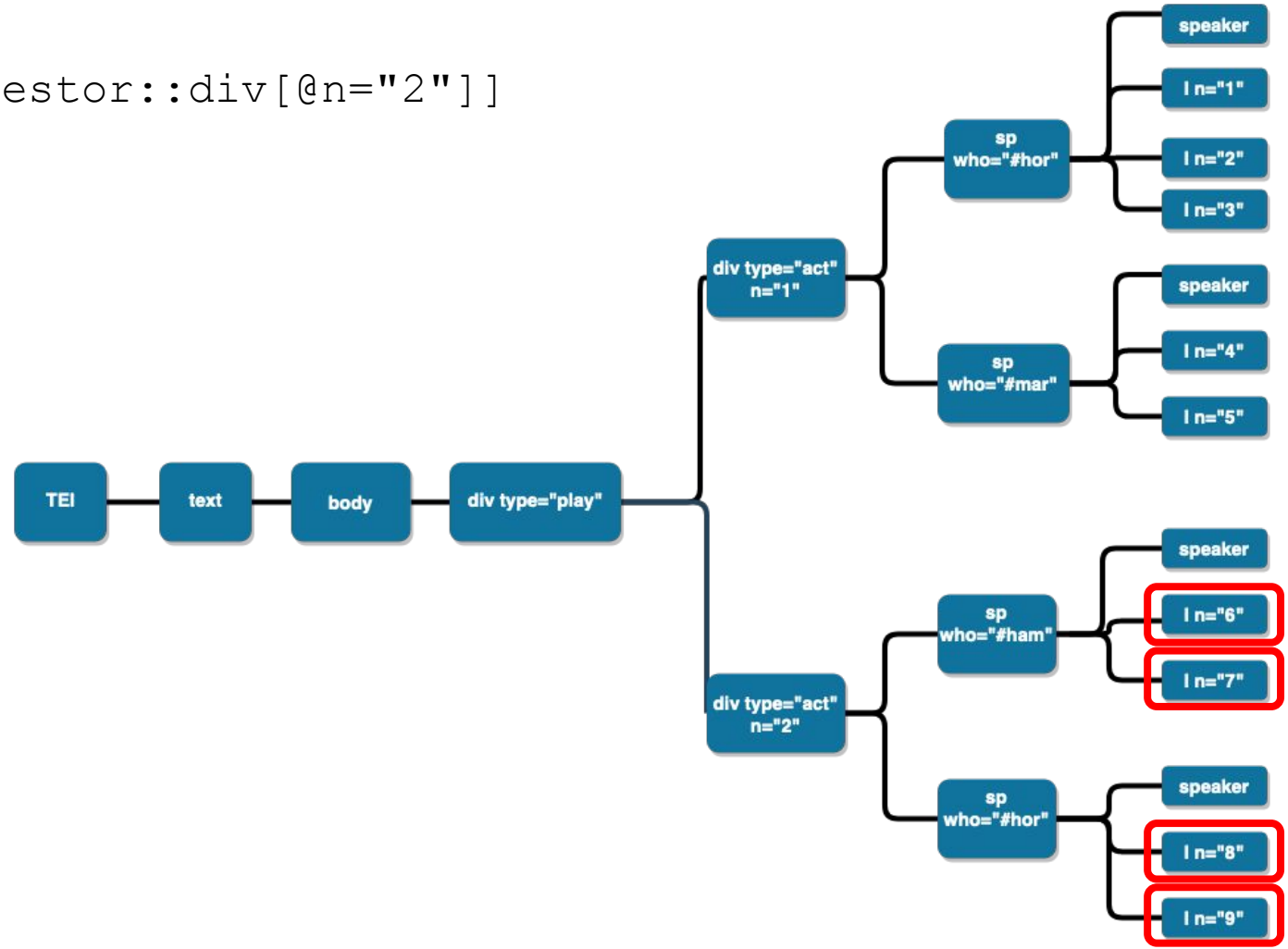
TEI/text/body/div/div/sp/l[@n=”3”]



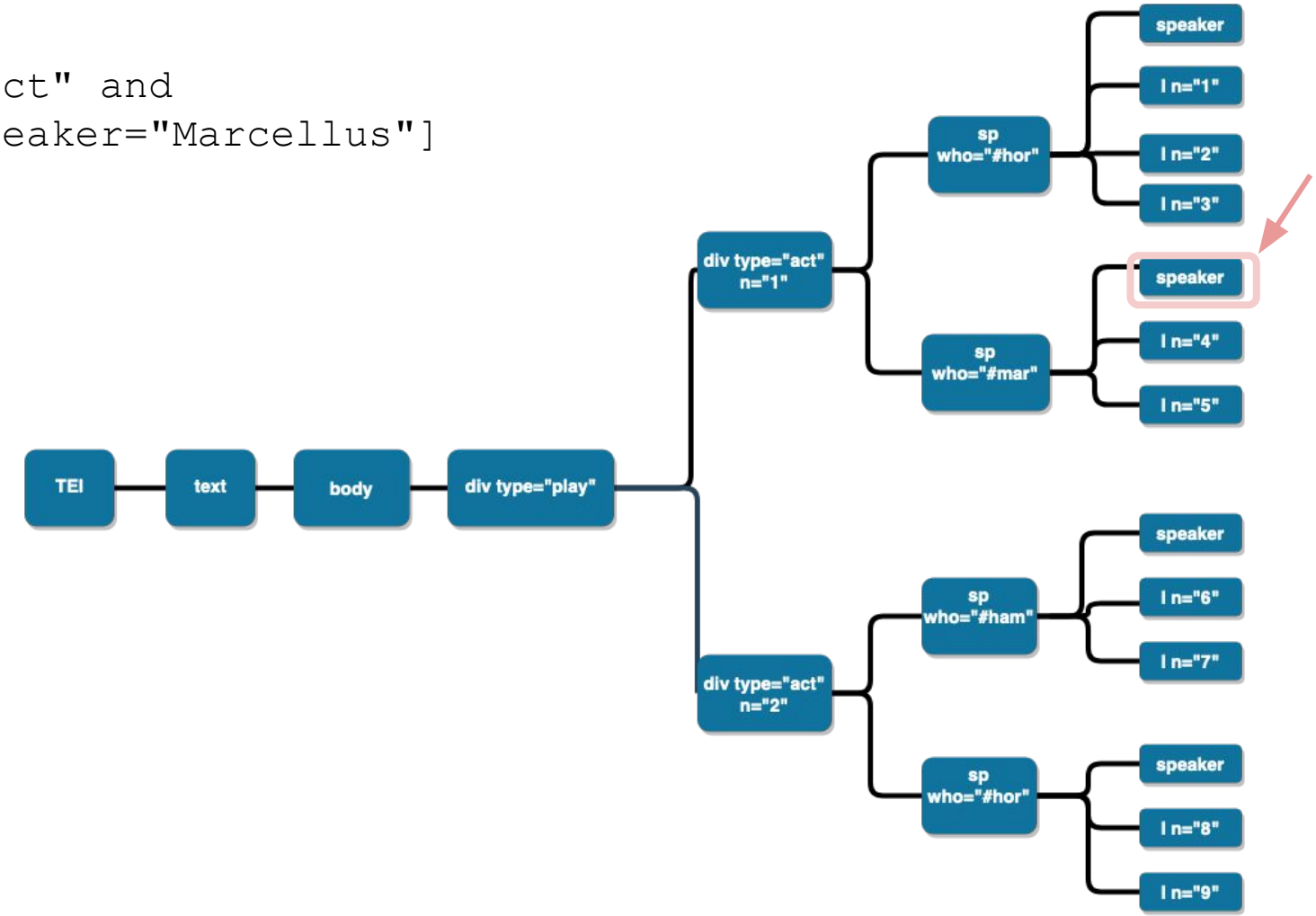
```
//l[ancestor::div[@n="2"]]
```



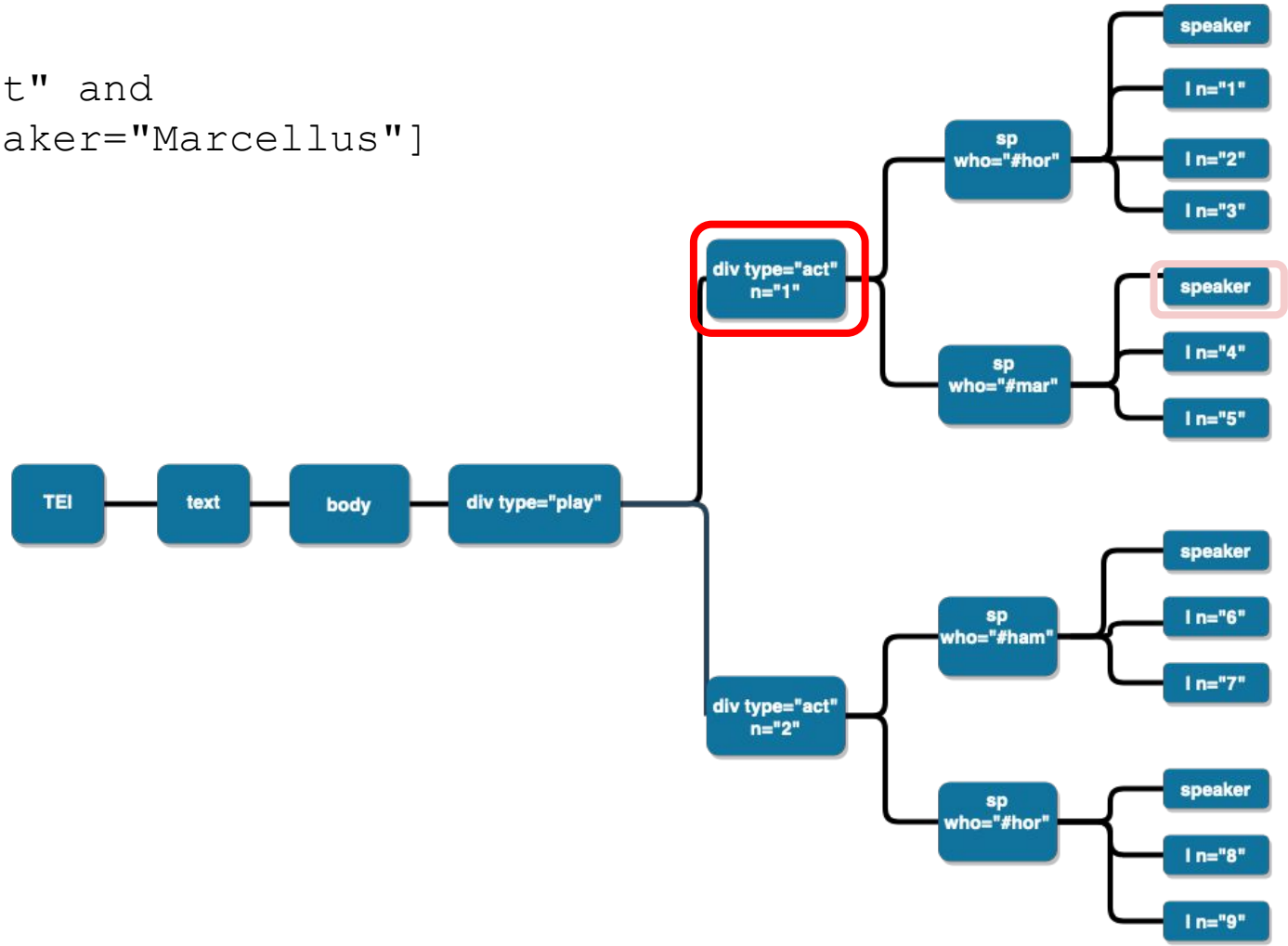
```
//l[ancestor::div[@n="2"]]
```



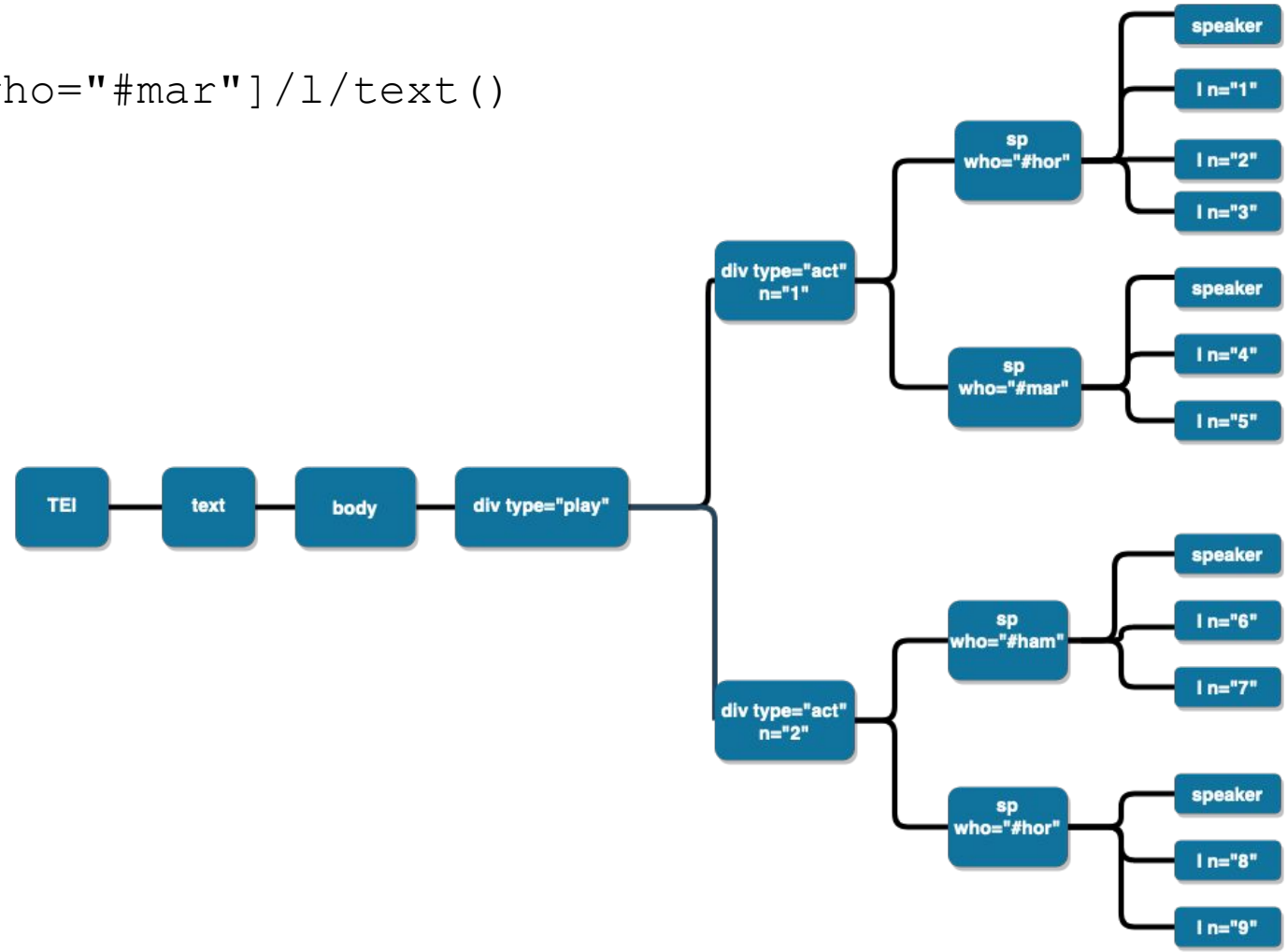
```
//div[@type="act" and
descendant::speaker="Marcellus"]
```



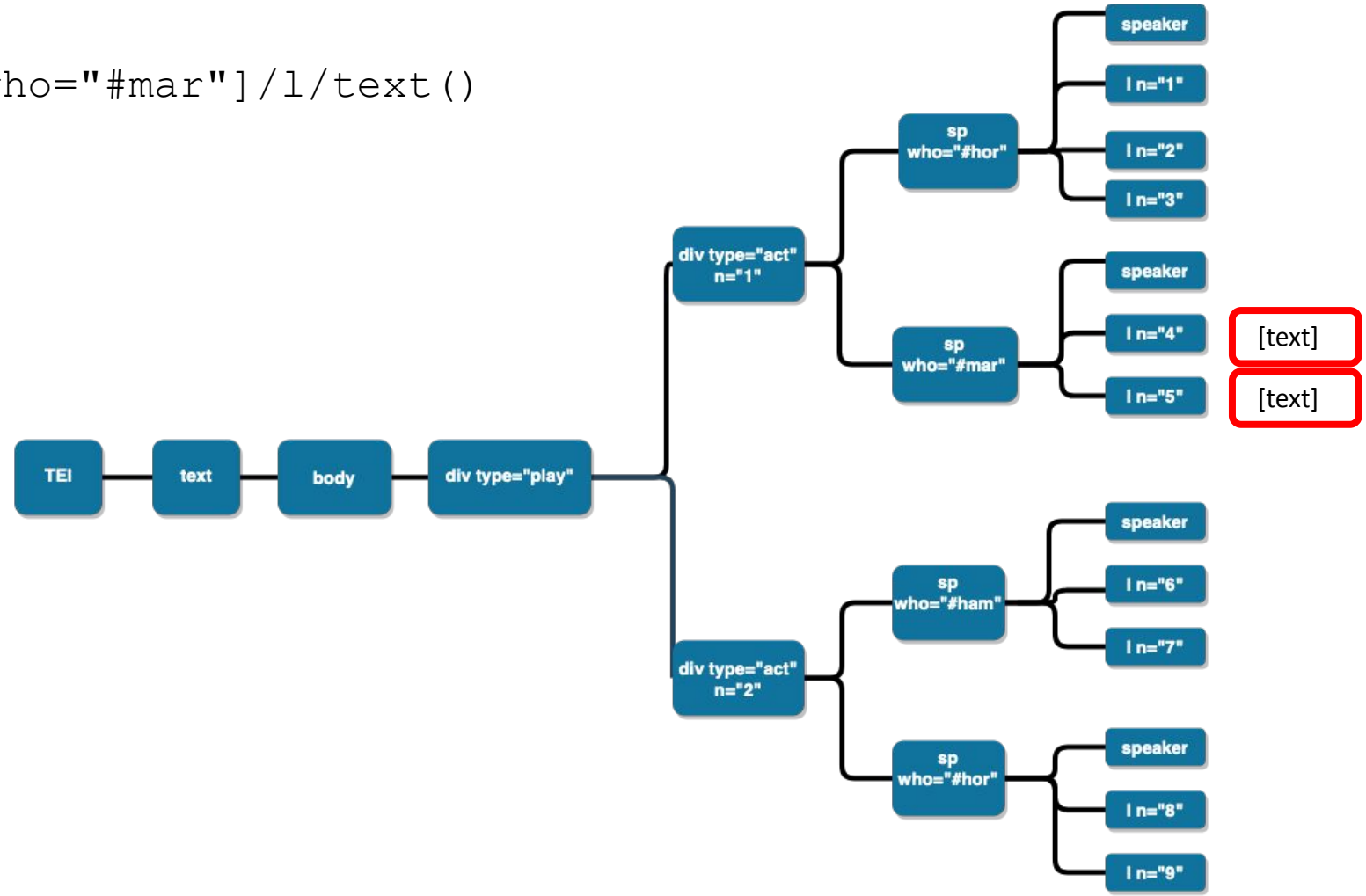
```
//div[@type="act" and
descendant::speaker="Marcellus"]
```



```
//sp[@who="#mar"]/l/text()
```



```
//sp[@who="#mar"]/l/text()
```



# 2. Getting acquainted with eXide

# eXide

- integrated development environment (IDE) of eXist for writing XQuery
- Start from your eXist dashboard
  - Start the eXist server
  - Go to localhost:8080 in your browser
  - Login as admin (no password)
  - Click on eXide

# 3. Navigate XML with XPath in eXide

# Exercises

1. Find all acts in Hamlet (`/db/apps/shakespeare-pm/data/F-ham.xml` in eXide)
2. Find all scenes in all acts
3. Find all stage directions that are part of a line: i.e, the stage direction that has an `<l>` as anchestor (=not direct parent)

# References and further reading


- DeRose, Steven J., David G. Durand, Elli Mylonas, and Allen H. Renear. “What is text, really?” *Journal of computing in higher education* 1.2 (1990): 3-26;
- Renear, Allen H.; Mylonas, Elli; Durand, David.. “Refining our Notion of What Text Really Is: The Problem of Overlapping Hierarchies.” *Research in Humanities Computing* (Nancy Ide and Susan Hockey, eds.), Oxford: Oxford University Press, 1996;
- Clifford B. Anderson and Joseph C. Wicentowski, *XQuery for Humanists*, Texas AM University Press, 2020, chapter 4 (get your copy!);
- Elisa Beshero Bondar's course at the DHSI 2022 on eXist-db and XQuery;
- David Birnbaum's course on XPath functions on his personal website;
- Demo of eXist site with documentation and tutorials;
- James Cummings’ introduction of XPath at the DiXiT camp in 2014.